commodore

za sva vremena

Dragan Tanaskoski
Stevan Milinković
Vladimir Janković

Mikro knjiga



Commodore
za sva vremena

Izdavač:
Samostalno izdanje grupe autora
Dragan Tanaskoski
Stevan Milinković
Vladimir Janković

recenzent:
Ladislav Rupnik

lektura i korektura:
Milena Lopušina
Rajko Vukčević

tehnički urednik:
Vladimir Janković

fotografija na koricama
Jovan Grujić

korice:
Darko Čonkić

crteži:
Svetozar Mostarlić

dipl. ing. Dragan Tanaskoski dipl. ing. Stevan Milinković
dipl. ing. Vladimir Janković

Commodore za sva vremena

Drugo izmenjeno i dopunjeno izdanje

Mikro knjiga

Commodore za sva vremena
YU ISBN 86–80003–02–6
UDK: 68.31–181.48

adresa izdavača:
Mikro knjiga, P.O.Box 75, 11090 Rakovica–Beograd

prvo izdanje 1986. god
drugo izdanje 1986. god. – tiraž 4000
foto slog: „Mladinska knjiga", Ljubljana
štampa: „Mileševo", Prijepolje

Beograd 1986.

Predgovor

„Commodore za sva vremena" je knjiga nastala sa ciljem da čitaoci dobiju na jednom mestu što više potrebnog materijala o primeni računara Komodor 64, njegovom programiranju i načinu rada.

Do cilja koji smo sebi postavili, bilo je potrebno više od godinu dana rada na samoj knjizi. Napisano je dvanaest poglavlja kroz koja smo nastojali da čitaoca upoznamo sa računarom, svim njegovim mogućnostima i njegovom praktičnom primenom. Redosled i sadržaj poglavlja treba da omoguće početnicima postupno ovladavanje računarskom tehnikom i Komodorom, a poznavaocima da pruže jasan pregled svega potrebnog za efikasan rad sa računarom.

Nadamo se da smo uspeli u ovim nastojanjima i da smo napisali knjigu koja će biti korisna vlasnicima računara Komodor.

Beograd, 1986 — *Autori*

1
Osnovni pojmovi o računarima

1. 1. RAČUNAR, HARDVER, ŞOFTVER

Računar je mašina koja može automatski da obrađuje veliki broj podataka. Razvoj računara od mehaničkih i elektromehaničkih do savremenih elektronskih mašina bio je uslovljen opštim tehnološkim napretkom.

Po fizičkoj osnovi rada računari se mogu podeliti na mehaničke, hidraulične, pneumatske, električne, elektronske i kombinovane.

Prema tipu veličina koje se obrađuju dele se na diskretne (digitalne), kontinualne (analogne) i hibridne.

Pneumatski računari se prvenstveno koriste u svemirskoj tehnici zbog imunosti na smetnje koje mogu poticati od velikog ubrzanja, raznih zračenja, kao i od elektromagnetskih i elektrostatičkih pojava. Uobičajena je primena pneumatskog i elektronskog računara u paralelnom radu, tako da se postiže izuzetno velika pouzdanost.

Osnovni delovi pneumatskih računara su komore i membrane, a ulazne i izlazne veličine su pritisci gasa, koji može biti i vazduh. Velika brzina reagovanja potiče iz samog principa rada, to jest od istovremene obrade svih ulaznih veličina, što znači da se istovremeni uticaj svih ulaznih pritisaka skoro trenutno odražava na pritisak koji se smatra izlaznom veličinom.

Analogni računari su najbrži računari i zbog toga se primenjuju pretežno u naučne svrhe kada je potrebno obaviti složena računanja za veoma kratko vreme. Obično se primenjuju u simuliranju i upravljanju procesima u realnom vremenu. Pod simulacijom se podrazumeva takav proces koji u pogledu brzine izvođenja potpuno odgovara stvarnom procesu. Praktična primena analognih računara je najčešća kod raketa sa samonavođenjem.

Osnovna komponenta analognih računara je operacioni pojačavač. Ulazne i izlazne veličine su električni napon. Otuda je jasno da se izlazni napon koji je u funkciji ulaznih napona skoro trenutno postavlja na izračunatu vrednost. Programiranje se obavlja prespajanjem pojedinih sklopova čime nije ostvarena velika fleksibilnost.

Digitalni računari operišu diskretnim brojnim vrednostima koje su najčešće u binarnom brojnom sistemu. Ulazne i izlazne veličine su brojevi koji su predstavljeni kombinacijama dva stanja električnog napona. Ta dva stanja su postojanje ili nepostojanje napona. Digitalni računar izvršava određene operacije na osnovu programa koji je sastavljen od naredbi predstavljenih takođe brojevima. Ta osobina obezbeđuje maksimalnu fleksibilnost u odnosu na prethodna dva tipa koji se grade za rešavanje određenih problema, posle čega se više ne mogu programirati. To je dovelo do toga da digitalni računari nailaze na najširu primenu.

Najveći broj velikih računara i svi kućni računari su digitalne mašine.

Hardver (engl. hardware) računara čine njegove fizičke komponente: električne i mehaničke, kao što su transformatori, kondenzatori, otpornici, tranzistori, integrisana kola, provodnici, prekidači, kutija i drugo.

Računaru je pomoću naredbi potrebno poručiti šta da uradi sa podacima koji su mu stavljeni na raspolaganje. Naredbe su takođe jedna vrsta podataka. Niz naredbi čini program. Računar obavlja zadatak izvršavajući naredbu po naredbu programa.

Program i podaci kojima računar raspolaže, nazivaju se softver (engl. software). Programi i podaci mogu biti ugrađeni u računaru, a mogu se nalaziti i van njega: na papiru, papirnoj traci, magnetskim trakama ili diskovima, kao i u samom programeru.

1. 2. BIT, BAJT, BINARNI BROJ

U računar se unosi više raznovrsnih podataka kao što su slova, znaci, brojevi itd. Sve te podatke je potrebno prevesti u jedinstven oblik radi lakše obrade i skladištenja.

Gledano sa fizičkog stanovišta rada računara najjednostavnije je ustanoviti da li na nekom mestu postoji određeni napon ili ne. Za računar je bitna informacija o dva moguća stanja: ima ili nema, što odgovara stavovima logičkog mišljenja „tačno" ili „pogrešno". Takva informacija, odnosno, podatak, koji može imati samo jednu od dve moguće vrednosti naziva se bit. Bit je cifra u binarnom brojnom sistemu, a ime je nastalo kao skraćenica od engleskog naziva Binary digit. Bit je jedinica mere za količinu informacija.

Stanje bita označeno jedinicom ili nulom odgovara postojanju ili odsustvu napona ili struje ili neke druge fizičke veličine.

Osam bita čini bajt. On može imati jednu od 256 različitih vrednosti (2^8) jer je to broj različitih kombinacija od osam cifara (bita) koje mogu biti nula ili jedinica.

Veće jedinice su 1Kbit, kilobit=1024 bita (2^{10}); 1Mbit, megabit=2^{20} bita; 1Kbajt, kilobajt=1024 bajta; 1Mbajt, megabajt=2^{20} bajta. U ovom slučaju nazivi „kilo" i „mega" se koriste, jer približno odgovaraju svojim normalno dodeljenim vrednostima.

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
1	1	0	1	0	0	1	0

bit najveće težine — bit najmanje težine

Sl. 1. 1. Simbolički prikaz jednog mogućeg stanja bajta

Niz od određenog broja bita (nula i jedinica) naziva se binarnim brojem. Svi podaci koji se unose u računar predstavljaju se, odnosno, koduju se pomoću binarnih brojeva i u takvom obliku ih računar pamti i obrađuje.

Obrada binarnih brojeva je u računaru zasnovana na Bulovoj algebri koju je George Boole razvio još 1854. godine. Tada je izložio metodu za simboličko izražavanje logičnih stavova čija se praktična upotreba i vrednost nisu ni naslućivali. Tek je 1938. godine Šanon (Claude Shannon) pokazao da se Bulova algebra može primeniti u analizi kola sa releima

1. 3. MIKROPROCESOR, MEMORIJA, PERIFERNE JEDINICE

Računar se sastoji od električnih kola koja na svojim ulazima razlikuju samo dve vrednosti napona, jedna odgovara nuli, a druga jedinici. Takođe na njihovim izlazima se mogu pojaviti samo te dve vrednosti. Takva kola se nazivaju logičkim ili digitalnim kolima. Svako od njih se sastoji od desetak pa do nekoliko hiljada tranzistora napravljenih na jednoj sili-

cijumskoj pločici dimenzija približno 4 X 4 mm. Takva pločica (čip) je zapakovana u plastič-no ili keramičko kućište na kome su ostavljeni izvodi za ulaze i izlaze. Na taj način je dobijeno integrisano kolo.

Mikroprocesor je integrisano kolo koje rukovodi radom računara pa se naziva i central-nom procesorskom jedinicom. On iz memorije redom čita kodove naredbi i izvršava ih jednu za drugom. Mikroprocesor se sastoji iz komandnog organa, izvršnog organa i nešto malo radne memorije. U Komodoru se nalazi mikroprocesor 6510 koji obrađuje informa-cije (naredbe i podatke) koji su predstavljeni sa osam bita (bajt). Takvi mikroprocesori nazivaju se osmobitnim.

U memoriji mikroračunara pamte se programi i podaci. Memorija se sastoji od inte-grisanih kola koja u sebi sadrže veliki broj memorijskih ćelija. Svaka memorijska ćelija se sastoji od jednog ili više tranzistora i može da pamti nulu ili jedinicu. Ćelije su organizova-ne tako da osam ćelija čini jednu celinu (bajt). Svaki bajt u memoriji je označen brojem koji se naziva adresom memorijske lokacije. Ona je potrebna zbog pristupanja željenom bajtu iz razloga upisivanja ili iščitavanja određenog podatka.

Postoji više tipova memorije koji se razlikuju u tehnologiji izrade i načinu rada. Dve najvažnije vrste memorija su RAM i ROM.

RAM memorija (engl. random access memory — memorija sa direktnim pristupom). U svaku adresiranu memorijsku lokaciju podatak se može upisati, a takođe se iz nje može i pročitati ono što je prethodno bilo upisano. Karakteristično je da se podaci prilikom nes-tanka napona napajanja nepovratno gube.

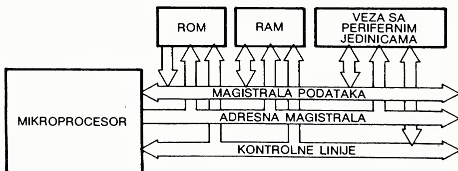
Iz ROM memorije (engl. read only memory) podaci se mogu samo čitati. Upisivanje podataka se obavlja u fabrici u kojoj se memorija i proizvodi. To znači da su podaci trajno zabeleženi i da se ne mogu izgubiti nestankom napona napajanja.

Praktična upotreba računara obavezno zahteva i upotrebu perifernih jedinica za raz-menu podataka između računara i korisnika. Za unošenje podataka se najčešće koristi tasta-tura. Za prikazivanje podataka i rezultata obrade, od perifernih jedinica se koriste TV ek-ran (monitor) i štampač. Za masovno pamćenje podataka na magnetofonsku traku, magnet-ni disk ili fleksibilni disk upotrebljavaju se kasetofon i disk jedinice.

Sve periferne jedinice se na računar priključuju preko električnih kola za vezu koja se nazivaju interfejsi (engl. interface).

Na slici je prikazana blok šema mikroračunara sa pojedinim sklopovima.

Pojedini delovi mikroračunara su povezani magistralom podataka, adresnom magi-stralom i kontrolnim linijama.



Sl. 1. 2. Tipičan sastav mikroračunara

Magistralu podataka (engl. data bus) čini 8 vodova preko kojih se podaci predstavljeni u obliku bajta prebacuju iz memorije ili periferne jedinice u mikroprocesor i obrnuto.

Adresna magistrala (engl. address bus) se sastoji od 16 vodova preko kojih se određuje (adresira) sa kojim memorijskim ćelijama ili perifernim jedinicama mikroprocesor razmenjuje podatke. Moguće je adresirati 65536 (2^{16}) različitih lokacija (adresa).

Preko kontrolnih linija se kontroliše i usmerava protok informacija. Tako se na primer preko kontrolnih linija određuje da li će se u memoriju nešto upisati ili iz nje pročitati.

1. 4. PROGRAMSKI JEZICI

Naredbe koje mikroprocesor izvršava smeštene su u memoriji u obliku binarnih brojeva. Tako predstavljene naredbe se nazivaju mašinskim naredbama ili mašinskim instrukcijama. Skup svih takvih naredbi koje koristi jedan mikroprocesor naziva se mašinski jezik. Jedna naredba u mašinskom jeziku određuje jednu elementarnu operaciju koju izvršava računar.

Program se može pisati i tako što se mašinske naredbe u obliku binarnih brojeva unose u računar. Ovako nepraktičan način pisanja programa je prevaziđen tako što je za svaku mašinsku naredbu odabran simbol koji se sastoji od nekoliko slova (to je najčešće skraćenica naziva naredbe na engleskom jeziku). Pojedine adrese i podaci takođe mogu imati svoja simbolička imena. Na ovaj način se došlo do simboličkog mašinskog jezika, odnosno asemblerskog jezika. Simbolički mašinski jezik se može svrstati između mašinskog jezika i viših programskih jezika. Program koji je napisan na simboličkom mašinskom jeziku treba prevesti na mašinski jezik. Taj posao može obavljati sam računar pomoću sistemskog programa koji se naziva asembler.

Viši programski jezici omogućuju daleko lakše programiranje. Programer ne mora dobro da poznaje konfiguraciju računara, a ne mora ni da vodi računa o elementarnim operacijama računara koje se obavljaju automatski. To je omogućeno time što je u okviru jedne naredbe objedinjeno više mašinskih naredbi.

Svi programi koji su pisani na nekom od viših programskih jezika mogu se izvršavati na dva načina. Prvi način je nešto komplikovaniji za upotrebu, ali je izvršavanje programa znatno brže. Taj način se sastoji u tome da se program napiše na nekom od viših programskih jezika i da se zatim upotrebi program koji se zove prevodilac ili kompajler (engl. compiler). On će izvršiti prevođenje napisanog programa na mašinski jezik. Izvršavanje ovako prevedenih programa je brzo jer se u toku izvršavanja naredbi ne vrši prevođenje, za razliku od drugog načina.

Drugi jednostavniji način za upotrebu se sreće u svim kućnim računarima. On se sastoji u tome da se prilikom startovanja nekog napisanog programa aktivira i program koji se naziva interpreter. Ovaj program se obično nalazi u ROM-u računara i on prevodi na mašinski jezik i automatski izvršava naredbu po naredbu. Ovaj način je znatno sporiji od prethodnog jer se prevođenje naredbi obavlja pri svakom njihovom izvršavanju. U Komodoru se nalazi bejzik interpreter, a mogu se nabaviti prevodioci za bejzik i druge programske jezike.

Viši programski jezici koji se najčešće koriste su: FORTRAN (formula translation), COBOL (common business oriented language), BASIC (beginner's all – purpose symbolic instruction code) i neki moderniji jezici kao što su PASCAL, C i ADA.

FORTRAN je namenjen matematičko numeričkim primenama gde su potrebna dugotrajna i složena računanja. Ulazno/izlazne operacije su skromnijih mogućnosti.

COBOL je namenjen poslovnim primenama gde se obrađuje veliki broj ulazno/izlaznih podataka. Zbog različitog profila korisnika kojima je kobol namenjen, definisane su naredbe na engleskom jeziku.

BASIC (bejzik) je jedan od najjednostavnijih programskih jezika koji se može koristiti u skoro svim oblastima (poslovne, naučne, tehničke itd.). Jednostavan je za upotrebu i ne zahteva profesionalno znanje programiranja.

Moderni jezici omogućavaju struktuirano programiranje i upotrebu struktuiranih podataka. Oni se najčešće koriste za pisanje sistemskih programa.

1. 5 KORISNIČKI I SISTEMSKI PROGRAMI

Računar se nabavlja sa namerom da se za nešto i koristi. Ber obzira da li je to za igranje, računanje, crtanje, obradu podataka, teksta ili neko automatsko upravljanje i merenje, mora se posedovati i odgovarajući program. Taj program je napravljen za određenu namenu i naziva se aplikativnim ili korisničkim programom. Ove programe korisnici mogu sami praviti, ali postoje i specijalizovane kuće za pravljenje programa namenjenih tržištu.

U računarski razvijenijim sredinama postoje i biblioteke programa iz kojih se oni mogu pozivati i koristiti. Članovima biblioteka je besplatno stavljena na raspolaganje određena količina često korišćenih i potrebnih programa, dok je za upotrebu nekih programa potrebna prilično velika novčana nadoknada.

Druga vrsta programa su sistemski programi čija je glavna uloga da omoguće korisniku što lakši i komotniji rad na računaru. Sistemski programi vode računa i omogućuju da se program i podaci upišu u računar, prikažu na ekranu, izmene, zapamte, da se program pusti u rad, zaustavi i drugo.

2
Primena računara Komodor 64

Komodor 64 je jedan od najrasprostranjenijih i najuniverzalnije primenjivanih računara. Do toga nije došlo slučajno, već je to posledica sistematičnog pristupa pri projektovanju računara. Tada su uzete u obzir ne samo trenutne potrebe korisnika računara već i buduće, do kojih će se dolaziti razvojem informatike, računarske tehnike i elektronike. Tako upotrebna vrednost Komodora 64 i dalje raste jer njegove mogućnosti još nisu potpuno iskorišćene.

Popularnost jednog kućnog računara se ogleda u njegovoj ceni, masovnosti, softverskoj podršci, mogućnosti priključenja dodatnih uređaja kao i u dostupnoj literaturi namenjenoj njegovoj upotrebi. Komodor 64 je za par godina svog postojanja potpuno ispunio sve ove zahteve, čime se može objasniti njegova velika i stalna aktuelnost.

Upotrebu računara Komodor moguće je najopštije podeliti na kućnu i poslovnu. U kućnu primenu spadaju aktivnosti koje se obavljaju radi zabave, informisanja ili učenja. Kućna upotreba je najdostupniji put da se široki krug korisnika upozna sa računarom i njegovim mogućnostima. U poslovnoj primeni od računara se zahteva pomoć u obavljanju raznih proračuna, obradi teksta i podataka, organizaciji neophodnih informacija i sl. Komodor je računar koji se može veoma uspešno primeniti i za zabavu i za povećanje efikasnosti poslovanja.

Igre

Upotreba Komodora u igrama je najrasprostranjenija. Tome doprinosi izvanredno veliki broj programa sa kvalitetnim igrama. Razne vrste igara (akcione, strategijske, simulacije, logičke) sa odličnim idejnim, grafičkim i zvučnim rešenjima svojom atraktivnošću objašnjavaju bezbrojne sate koje pojedinci provode pored računara.

Muzika

Od svih računara Komodor 64 je najčešće primenjivan u muzici. Tome je doprinelo postojanje audio sintesajzera i pojava standarda za povezivanje elektronskih muzičkih instrumenata. Preko MIDI (musical instruments digital interface) interfejsa je omogućeno povezivanje računara sa sintesajzerima, orguljama, elektronskim bubnjevima kao i njihovo međusobno povezivanje.

Komodor u sebi sadrži sintesajzer sa tri audio kanala (oscilatora). To omogućuje višeglasno sviranje i upotrebu računara kao pravog muzičkog instrumetna. U sprezi sa nekim drugim sintesajzerom Komodor se može iskoristiti da upravlja njegovim radom čime se najčešće dobija vrlo dobar sekvencer. To je uređaj koji će moći da odsvira jednom programiranu melodiju.

Matematička tehnička primena

Rešavanje matematičkih, fizičkih, tehničkih i organizacionih problema u kući, školi ili na fakultetu može biti kreativnije i celishodnije uz pomoć računara.

Pri radu na rešavanju konkretnih zadataka pomoću računara programer je u situaciji da mora dobro da razume problem, da razmišlja na uopšten način, da sagleda ograničenja i specijalne slučajeve i na kraju da napravi program koji će dovesti do rezultata.

Ovakav način zbližavanja sa računarom dovodi do toga da se već u procesu razmišljanja o problemu, čovek oslanja na računar i pokušava da problem reši pomoću njega.

Ponekad, rešavanje zadatka pomoću računara oduzme više vremena nego da se radi bez njega. U slučaju da se predviđa da će se bar još jedanput raditi isti ili sličan zadatak, potrebno je prihvatiti pomoć računara. U tom slučaju će uštede u vremenu biti znatno veće, a novostečeno iskustvo će se iskoristiti pri rešavanju većih problema.

Praksa je pokazala da čak i za neka jednostavnija izračunavanja gde se operiše sa većim brojem bodataka, treba praviti program. Razlog je jednostavan. Na osnovu jednom unetih podataka moguće je izvršiti sva računanja, mogu se lako prikazati i korisni međurezultati, a konačni rezultati se mogu tabelirati i tako sređeni dobiti na štampaču. U slučaju upotrebe džepnog računara bilo bi potrebno iste ulazne podatke unositi više puta, a rezultati ne bi bili trajno zabeleženi.

Svako ko je bar jedanput uradio kompletan proračun, na primer za motanje transformatora, zna koliki posao treba ponoviti za nov proračun sa izmenjenim zahtevima. Vreme utrošeno za proračun tri transformatora je dovoljno da se napravi program sa svim potrebnim tabelama. Tada će svaki proračun da se obavi brzo, a dobiće se i podaci koji su se obično uzimali približnim zbog velikog obima računanja.

Za Komodor su napisani mnogobrojni programi za različite matematičke i tehničke pripreme. Nabavkom odgovarajućeg programa korisnik računara može naći rešenje za svoj problem bez potrebe za razvijanjem sopstvenog programa.

Obrada teksta

Najčešća praktična primena kućnih računara je pri pisanju. Računar sa disk jedinicom i štampačem predstavlja daleko efikasnije sredstvo za rad od najbolje pisaće mašine.

Suštinska razlika u odnosu na pisaću mašinu je u tome što se tekst pre konačnog štampanja može složiti u željeni oblik, lako modifikovati i korigovati, kao i sačuvati za kasniju ponovnu upotrebu.

Za pisanje teksta pomoću računara najčešće se koriste posebni programi koji se obično zovu tekst procesori ili programi za obradu teksta. Za Komodor postoji više ovakvih programa koji su međusobno vrlo slični.

Tekst procesori omogućuju da se tekst upiše u željenom formatu i smesti u računarsku memoriju, da se modifikuje, snimi na disketu i odštampa na štampaču. Pisanje teksta je jednostavno, a sve uočene greške se mogu lako ispraviti brisanjem pojedinih slova, cele reči ili reda, kao i njihovim umetanjem u tekst.

U toku pisanja je moguće određivati tip slova kojima će tekst biti odštampan kao i to da li je deo teksta podvučen ili istaknut dvostrukim štampanjem. Takođe se mogu postaviti leva i desna margina i omogućiti da tekst bude poravnat po desnoj margini. Tada ne treba voditi računa o prenosu u nov red jer to obavlja sam računar. Deljenje teksta na stranice kao i određivanje proreda može se obaviti neposredno pre štampanja.

Baze podataka

Baze podataka su specijalno organizovani skupovi srodnih podataka za čiju efikasnu manipulaciju postoje programi koji se nazivaju sistemi za upravljanje bazom podataka. Primenjuju se na većim računarima mada se mogu koristiti i na nivou kućnih računara.

Za Komodor postoje opšti i specijalizovani programi za rad sa bazama podataka.

Na primeru ditribucije nekog proizvoda biće opisano kako se koristi baza podataka.

Oformljena je datoteka „PRODAJA" u koju su smešteni podaci o prodajnim mestima datog proizvoda. Svi podaci o jednom prodajnom mestu se nalaze u datoteci u delu koji se naziva slog (engl. record). Svaki slog ima redni broj koji je ujedno i broj prodajnog mesta. Slog se sastoji od polja. Svako polje je definisani prostor u kome se sadrži jedan podatak. Svakom polju se dodeljuje naziv, broj znakova predviđen za smeštanje odgovarajućeg podatka kao i tip podatka (broj ili karakteri). U ovom konkretnom slučaju su određena polja sa sledećom strukturom: datum, naziv, adresa, pošt. broj, grad, količina, iznos i komentar. Za datum, pošt. broj, količinu i iznos je predviđeno da budu samo numerički podaci. Novoprispeli podaci o prodaji se naknadno unose u računar.

Na osnovu unetih podataka jednostavno se ostvaruje štampanje adrese, utvrđuje potražnje i prodaje, uticaj reklame i još niz podataka interesantnih za evidenciju i statistiku.

Povezivanje u mrežu

Sa povećanjem broja računara javlja se logična potreba njihovog povezivanja u mrežu.

Računarska mreža se obično formira kada se stvori računarski centar u kome se nalazi neki snažniji računar sa zadatkom da većem broju korisnika stavi na raspolaganje biblioteku programa i razne baze podataka.

Računarska mreža može biti oformljena i bez centralnog računara. Dovoljno je da se bar dva računara povežu i da se ostvari uzajamna razmena informacija.

Povezivanje u mrežu ostvaruje se preko telefonskih linija i uređaja koji se zovu modemi. Modem obezbeđuje transformaciju signala iz računara u oblik koji je pogodan za prenos preko telefonskih linija kao i obratno.

Za Komodor postoje dva tipa modema. Jednostavniji tip koristi i telefonski aparat kao deo sistema. On obezbeđuje pretvaranje električnih signala iz računara u zvuk koji se dalje prenosi telefonom. Drugi bolji, pouzdaniji i skuplji metod se zasniva na direktnom uključenju modema u telefonsku liniju.

Edukativna primena

Upotreba računara u obrazovanju je i pored svih pogodnosti nastalih u poslednje vreme relativno zapostavljena oblast.

Tekstualna informacija sa pratećim vizuelnim i zvučnim efektima omogućuje razumevanje i prihvatanje novih pojmova i informacija kao i odnosa među njima.

Najbitniji elementi koji doprinose efikasnosti načina prezentiranja nove materije su stvoreni model i animacija slike.

Novi pojmovi, pojave i zakonitosti se izlažu preko modela koji uspostavlja analogne veze između već poznatih i novih nepoznatih elemenata. U gradnji modela računar omogućuje da do izražaja dođe puna kreativnost autora metodske jedinice jer su izbegnuta sva ograničenja fizičke realizacije. To znači da se već u samom modelu može dozvoliti izvestan stepen imaginacije koji pozitivno deluje na učenika jer mu takav pristup podstiče razmišljanje u željenom smeru.

Animirani crtež i zvuk predstavljaju bitnu sponu preko koje se pospešuje proces pamćenja. Iz tog razloga u animaciji i zvuku mora da postoji nekoliko ključnih prepoznatljivih elemenata koji su sinhronizovani sa bitnim momentima i informacijama.

Za proveru i utvrđivanje novostečnog znanja računar pruža izuzetne mogućnosti u obrazovanju. Zahvaljujući interaktivnom odnosu između učenika i računara postignuto je da se učenik tim aktivnim kontaktom preko pitanja, odgovora, slike i zvuka više povezuje sa materijom koju uči nego u slučaju pasivnog slušanja ili čitanja.

Pravljenje obrazovnih (edukativnih) programa na računaru, pored računarskog znanja, zahteva i dobro poznavanje pedagogije, psihologije učenja i metodike. Zbog toga i nije čudo što je za sada retka pojava nekog boljeg programa u ovoj oblasti.

3
Uvod u rad sa Komodorom

3.1 SASTAV MIKRORAČUNARSKOG SISTEMA KOMODORA

Mikroračunarski sistem Komodora 64 sastoji se najčešće od mikroračunara Komodor, mrežnog napajanja (ispravljača), televizijskog prijemnika i kasetofona.

Komodor se sastoji od tastature i štampane ploče sa elektronskim komponentama.

Tastatura služi za komandovanje računarom i unošenje programa i podataka. Ispravljač obezbeđuje jednosmerni napon potreban za napajanje računara. Podaci, programi, poruke, crteži i drugo prikazuju se na TV ekranu u boji. Komodor može proizvoditi tonove i melodiju koji se reprodukuju na zvučniku televizora. Kasetofon služi da se na magnetnoj traci (kaseti) zabeleže podaci i program, a i da se sa nje prebace u Komodor.

Ova osnovna konfiguracija je najčešće dopunjena štampačem i disk jedinicom. Štampač omogućuje da Komodor ispisuje podatke i programe na papiru, što otvara mogućnosti poslovne primene Komodora. Disk jedinica, kao i kasetofon, omogućuje snimanje podataka i programa za kasniju upotrebu. Njegova prednost je u bržem pristupu podacima i programima, što ga čini nezamenijivim za primene u kojima je efikasnost prioritetna.

Veoma čest deo u sistemu Komodora je palica za igru (džojstik). Ovladavanje velikim brojem akcionih igara je olakšano korišćenjem palice umesto tastature.

Mikroračunarski sistem Komodora može se proširiti i drugim uređajima kao što su: svetlosna olovka, grafička tabla (za brže crtanje), modem (za vezu sa drugim računarima preko telefonske linije), sintesajzer (za veće muzičke mogućnosti), itd. Ovi uređaji se nešto ređe primenjuju.

Jedan od uređaja mikroračunarskog sistema koji zaslužuje veću pažnju je monitor. Monitor je specijalno napravljen uređaj koji zamenjuje televizor. Priključen na računar daje visoko kvalitetnu, jasnu i mirnu sliku uz manje štetnog zračenja ekrana. U primenama računara u kojima se zahteva dugotrajno gledanje ekrana, upotreba monitora nije stvar konfora, već nužnosti.

3.2 PUŠTANJE U RAD

Minimalna konfiguracija potrebna za rad sastoji se od Komodora 64, ispravljača i TV prijemnika.

Komodor se povezuje sa televizorom preko kabla dobijenog uz računar. Jedan njegov kraj se uključi u standardni antenski priključak na zadnjoj strani Komodora, a drugi u antenski priključak televizora. Ispravljač se uključi u gradsku mrežu (220V) i poveže sa Komodorom

preko priključka POWER na njegovoj bočnoj strani. Pritiskom na taster ON, koji se nalazi pored priključka POWER, Komodor je uključen.

Da bi se na ekranu pojavila slika koju stvara računar, potrebno je televizor podesiti na 36. kanal (UHF opseg). Sa radom se može početi kada se na ekranu pojavi poruka:

**** COMODORE 64 BASIC V2 ****

64K RAM SYSTEM 38911 BASIC BYTES FREE

READY.

Ovom porukom javlja se da je računar spreman za rad, da prihvata naredbe bejzik programskog jezika i da je od ukupno 64K RAM memorije za korišćenje u bejziku na raspolaganju 38911 bajta.

Dobijenu sliku regulacijama osvetljenja, kontrasta i biranja kanala treba podesiti za što prijatniji rad. Korišćenjem televizora ili monitora u boji pokazaće se da je dobijena slika tamno plava sa svetloplavim okvirom i svetloplavim slovima. Ukoliko se slika nije pojavila potrebno je proveriti ponovo da li je sve priključeno kako treba i da li je TV prijemnik dobro podešen.

Priključivanje drugih uređaja na Komodor treba obavljati samo kada su i Komodor i uređaji koji se priključuju isključeni (disk jedinica, štampač...).

3.3 TV EKRAN

Računar piše i crta po središnjem pravougaonom delu ekrana. U njemu se može ispisati 25 redova sa po 40 karaktera (slova, brojeva i raznih drugih znakova). Središnjem delu se može zadati jedna od raspoloživih boja. Karakteri se mogu prikazivati u raznim bojama. Obodonom delu ekrana, okviru (engl. border) može se samo promeniti boja.

Po uključivanju računara, ispod početne poruke, pojavljuje se kvadratno polje veličine jednog karaktera koje trepće. To je pokazivač ili kursor (engl. cursor). On pokazuje na kom mestu će se ispisivati ili brisati karakteri prilikom pritiskanja tastera. Odgovarajućim komandama može se pomerati po celom ekranu omogućavajući pisanje i brisanje na željenom mestu.

3.4 RAD SA TASTATUROM

Komodor ima 66 tastera, od kojih većina ima više funkcija. Osnovna namena tastera je ispisana na njegovoj gornjoj površini. To su najčešće slova i brojevi. Njihovim pritiskanjem na ekranu će se ispisivati odgovarajuća slova, brojevi ili znaci.

U gornjem redu tastera nalaze se tasteri sa brojevima. Njihovim pritiskanjem na ekranu se ispisuju brojevi. Znaci na tasterima koji se nalaze iznad brojeva ispisuju se na ekranu ako je za vreme pritiskanja tastera pritisnut i taster SHIFT. Taster SHIFT se nalazi na dva mesta, u levom i desnom delu donjeg reda tastera. On odgovara tasteru za prelazak sa malih na velika slova kod pisaće mašine.

Primer: Ako se pritisne taster na kome je cifra **1** i znak !, na ekranu se ispisuje broj **1**, a ako je pri tome pritisnut i taster SHIFT na ekranu se ispisuje znak uzvika.

Pritiskanjem tastera sa slovima na ekranu se ispisuju slova, a ako je jednovremeno pritisnut i taster SHIFT na ekranu se ispisuju desni grafički simboli, koji se nalaze sa donje strane tastera.

Primer: Dok je pritisnut SHIFT, pritiskom na taster S na ekranu se ispisuje grafički simbol srca.

Grafički simboli koji se nalaze na levoj strani donjeg dela tastera dobijaju se uz pritisnut taster sa Komodorovim znakom. To je taster koji se nalazi prvi sleva u četvrtom redu (levo od tastera SHIFT). Taj taster se zove Komodor taster i u daljem tekstu će se označavati sa C=.

Izloženo o dobijanju grafičkih znakova važi samo ako Komodor radi sa prvim skupom Karaktera. U prvom skupu Komodor radi sa velikim slovima i svim grafičkim simbolima. Prelazak na drugi skup se ostvaruje jednovremenim pritiskom Komodor tastera i tastera SHIFT. Tada Komodor radi sa malim i velikim slovima i jednim delom grafičkih karaktera. U drugom setu velika slova se dobijaju pritiskanjem tastera SHIFT i željenog slovnog tastera. Pritiskom Komodor tastera i slovnog tastera, isto kao u prvom skupu, dobija se grafički simbol.

Povratak na prvi skup karaktera ostvaruje se isto kao i prelazak na drugi skup, jednovremenim pritiskom tastera Komodor i SHIFT.

U slučaju pisanja kada je potrebno taster SHIFT držati dugotrajno pritisnut može se upotrebiti taster SHIFT LOCK, koji se nalazi iznad levog tastera SHIFT.

Pomeranje kursora po ekranu bez ispisivanja se ostvaruje tasterima CRSR. Postoje dva takva tastera i nalaze se u krajnje desno u četvrtom redu tastera. Oni se razlikuju po dejstvu. Taster CRSR na kome su nacrtane strelice za gore i dole pomeraće kursor nadole. Ako je jednovremeno pritisnut i taster SHIFT kursor će se pomerati nagore. Drugi CRSR taster, na kome su nacrtane strelice za levo i desno pomeraće kursor nadesno, a ako je pritisnut i taster SHIFT kursor će se pomerati nalevo. Tasteri za pomeranje kursora automatski ponavljaju svoju funkciju dok god su pritisnuti.

Taster praznog polja, u donjem redu, je najveći taster na tastaturi. Služi za ispisivanje praznog karaktera, karaktera praznog polja. Odgovara tasteru razmaknici na pisaćim mašinama.

Brisanje (engl. delete) ispisanih karaktera se ostvaruje pritiskom tastera INST/DEL. Pri tom će biti obrisano ono što je napisano levo od pokazivača (kursora). Isti taj taster se koristi i za umetanje slova unutar reči (engl. insert). To se ostvaruje na sledeći način: Potrebno je dovesti kursor na mesto na koje se žele umetnuti karakteri i pritisnuti taster INST/DEL, ali uz pritisnuti taster SHIFT. Ispisani karakteri desno od kursora će se pomerati u desno stvarajući prostor za upisivanje teksta. Ova funkcija se takođe automatski ponavlja sve dok su tasteri pritisnuti.

Taster CLR/HOME se nalazi na desnoj strani gornjeg rada tastera. Njegovim pritiskom kursor će se premestiti u gornji levi ugao ekrana, bez obzira gde se nalazi. Ako se pritisne uz jednovremeno pritisnut i taster SHIFT kompletni sadržaj ekrana će biti obrisan, a kursor premešten u gornji levi ugao ekrana.

Taster CTRL služi za zadavanje boja i inverznog načina ispisivanja karaktera. Taj taster se koristi samo uz tastere sa brojevima. Na donjoj strani tih tastera se nalaze ispisane boje karaktera (spisak boja je dat u poglavlju 4 u naredbi PRINT) koje se dobijaju pritiskom tastera CTRL i tastera sa brojem tj. bojom.

Primer: Jednovremenim pritiskom tastera CTRL i tastera 1 kursor postaje crne boje (engl. black). Takođe i karakteri koji će se ispisivati biće crne boje.

Upotrebom tastera CTRL dobija se 8 boja. Komodor raspolaže sa još 8 boja koje se dobijaju istim postupkom samo što se umesto tastera CTRL koristi Komodor taster (C=). Tasterma CTRL i 9 prelazi se u inverzni (RVS ON) način ispisivanja karaktera na ekranu.

U tom načinu boja samog karaktera je zamenjena sa bojom osnove na kojoj je karakter nacrtan. Povratak u normalni, neinverzni (RVS OFF) način rada ostvaruje se pritiskom tastera CTRL i 0.

Funkcijski tasteri su četiri veća tastera na desnoj polovini računara. Njihovim pritiskanjem se ostvaruju funkcije tastera f1, f3, f5 i f7. Ako su pritisnuti uz pritisnuti taster SHIFT dobijaju se funkcije tastera f2, f4, f6 i f8. Po uključenju računara funkcijskim tasterima nije dodeljena nikakva funkcija i njihovo pritiskanje ne daje nikakav efekat. Oni su ostavljeni na raspolaganje korisniku računara da im dodeli neku namenu.

Taster RUN/STOP služi za prekidanje izvršavanja programa napisanih u programskom jeziku bejziku. O tome će biti više reči u narednom tekstu.

Taster RESTORE se koristi zajedno sa tasterom RUN/STOP. Njihovim pritiskom računar se dovodi u početno stanje. Određena unutrašnja stanja računara su dovedena u stanje po uključenju računara, boje na ekranu su se vratile na početne. Ovim tasterima se prekida svaki program pisan u bejziku i neki programi pisani na mašinskom jeziku.

Taster RETURN je ostavljen za kraj ovoga pregleda tastature. To je najznačajniji taster. Njegovim pritiskom se unose prethodno ispisani podaci u računar. U tekstu koji sledi biće detaljno prikazano njegovo dejstvo.

3.5 NAČINI RADA

Komodor može da radi na dva načina. Prvi je direktni ili kalkulatorski (engl. calculator mode) način rada, a drugi je programski način rada.

Direktni način rada

Kada se računar koristi direktno ispisuje se naredba, a zatim se pritiska taster RETURN. Tada računar odmah izvršava tu naredbu. To ne mora biti samo jedna već može biti i više naredbi koje moraju biti međusobno odvojene sa dve tačke.

Primer: PRINT "KOMODOR": PRINT "64"

Pritiskanjem odgovarajućih tastera treba na ekranu ispisati dati primer. Kada je to postignuto pritiskom na taster RETURN ispisaće se zadata reč **KOMODOR**, a ispod nje broj 64. Dva reda niže će biti ispisana poruka **READY.** koja označava da je računar izvršio ono što mu je zadato i da je spreman (engl. ready) za dalji rad.

Direktni način rada je pogodan za brza i kratka računanja u slučajevima da se rezultat neće dalje koristiti u računaru.

Primer: Ako se napiše: **PRINT ((2.31+2*7.8)/ ↑2**

i posle toga pritisne RETURN, Komodor će izračunati ovaj izraz i napisati rezultat 35.6409001

Napomena: U engleskom načinu označavanja brojeva ne koristi se decimalni zarez već decimalna tačka, pa 2.31 odgovara našem 2,31.

Simboli upotrebljeni u prethodnom primeru su:

+ simbol za sabiranje
* simbol za množenje
/ simbol za deljenje
↑ simbol za stepenovanje

Za direktni način rada važno je znati da računar ne pamti naredbe nakon njihovog izvršavanja.

Programski način rada

U programskom načinu rada prvo se piše program, a zatim se izvršava naredbom **RUN** i **RETURN.**

Program se sastoji od niza programskih linija. Svaka počinje brojem linije koji može da bude ceo broj od 1 do 63999. Za njima slede jedna ili više naredbi. Programska linija se ispisuje na ekranu, a zatim se pritiskom na taster RETURN ostvaruje da računar tu liniju zapamti. Pritisak na taster RETURN je obavezan i pri tome se naredbe u liniji neće izvršavati. Kursor će se po pritisku tastera spustiti na početak jednog reda niže, tako da se može pristupiti pisanju nove programske inije.

Da bi se olakšalo ispravljanje programa i umetanje novih programskih linija, brojevi linija ne bi trebalo da budu uzastopni brojevi. Uobičajeno je da se razlikuje za 10.

Primer: Napisati sledeće programske linije:

10 PRINT "COMMODORE" ⟨RETURN⟩
30 PRINT 64 ⟨RETURN⟩
20 PRINT 2+3 ⟨RETURN⟩

Linije treba napisati na sledeći način: Kursor tj. pokazivač treba dovesti na početak praznog reda. Pritiskanjem odgovarajućih tastera napisati **10 PRINT "COMMODORE"**, a zatim pritisnuti taster RETURN, što je u primeru označeno sa ⟨RETURN⟩. Kao što je rečeno to je obavezno da bi računar prihvatio napisanu liniju. Zatim se piše linija 30, pritiska taster RETURN, pa linija 20 i ponovo taster RETURN.

Prilikom ispisivanja programskih linija važno je da na ekranu red u kome se ispisuje linija bude prazan. Neki drugi karakteri u redu će ući u sastav programske linije, što će poremetiti željeni rad računara.

Linije nisu unešene u računar po rastućim brojevima linija. To nije obavezno jer će računar sam da ih poređa po rastućem poretku, što je prikazano u tekstu koji odmah sledi.

LIST

Od računara se može zatražiti da prikaže, odnosno izlista upisani program. To se postiže naredbom **LIST.** Ako iza naredbe ne stoji ni jedan broj, listanje počinje od linije sa najmanjim brojem. Ako je naveden broj iza naredbe prikazaće se samo linija sa tim brojem.

Primer: Ako se napiše:

LIST

i pritisne taster RETURN računar će prikazati ceo program. Ako se upiše:

LIST 20

i pritisne taster RETURN biće prikazana samo linija 20.

Po završenom listanju ispisaće se poruka **READY.**, a pokazivač će preći u red ispod poruke.

Ako se u računaru ne nalazi program ili željena linija računar neće imati šta izlistati i samo će napisati poruku **READY.** Ako je unešen prethodni primer od tri programske linije, izvršenjem naredbe **LIST** videće se da je računar linije poređao po brojevima linija.

U slučaju dužih programa prikazivanje programa se može usporiti pritiskom na taster CTRL, a prekinuti pritiskom na taster RUN/STOP.

RUN

Kada se želi da računar izvrši program koji je njemu, ispisuje se naredba **RUN**, a zatim se pritiska taster RETURN.

Primer: Upisati prethodno dat program od tri programske linije, a zatim napisati RUN i pritisnuti taster RETURN. Računar će ispisati na ekranu sledeće:

COMMODORE
64
5

Izvršenjem naredbe **RUN** program se izvršava od programske linije sa najmanjim brojem. Prvo će se ispisati reč **COMMODORE**, zatim broj 64, a na kraju će Komodor izračunati zbir brojeva 2 i 3 i napisati rezultat 5.

Ako se iza naredbe RUN navede broj, program će se početi izvršavati od linije sa tim brojem.

Primer: Upisati program:

```
10 PRINT CHR$(147)
20 PRINT 23*17
```

i izvršiti naredbu **RUN 20.** Računar će izračunati koliko je 23 puta 17 (oznaka za množenje je zvezdica) i napisati rezultat. Ako se izvrši naredba **RUN** program će se izvršavati od linije 10. To će dovesti do brisanja sadržaja ekrana (linija 10), i ispisivanje rezultata 391 (linija 20).

U ovom primeru nije naznačeno, a takođe ni u daljem tekstu, da je potrebno pritiskati taster RETURN po ispisivanju programskih linija u programskom načinu rada i po ispisivanju naredbi u direktnom načinu rada.

NEW

Program se briše iz memorije računara pomoću naredbe NEW.

Primer: Posle izvršenja naredbe

NEW

kompletan program će biti obrisan iz memorije računara. Ako se izvrši naredba

LIST

vidi se da nema šta da se izlista.

RUN/STOP

Pritiskom na taster RUN/STOP prekida se izvršavanje programa.

Primer: Upisati program:

```
10 PRINT "COMMODORE"
20 GOTO 10
```

Program startovati sa uobičajenim **RUN** i RETURN. Program će početi sa izvršavanjem od linije 10, napisaće se reč **COMMODORE**, zatim će preći na izvršavanje linije 20 koja će ga vratiti na liniju 10 i postupak će se ponavljati neprekidno. To se dešava veoma brzo i ekran će biti popunjen natpisom **COMMODORE**. Iako se posle toga prividno ništa ne dešava program se i dalje izvršava i ispisuje reč **COMMODORE**, pomerajući prethodno ispisani red naviše. Pritisak na tastere neće imati nikakvog efekta.

Pritiskom tastera RUN/STOP izvršavanje programa će biti prekinuto. Takođe će biti ispisana poruka o tome na kom mestu programa (broj linije) je došlo do prekida (poruka **BREAK**).

CONT

Prekinuti program će nastaviti da se izvršava ako se izvrši naredba CONT (engl. continue).

Primer: Program iz prethodnog primera će nastaviti da se izvršava ako se napiše CONT i pritisne taster RETURN.

Ekranski editor

Ispisivanje naredbi i programskih linija se može izvršavati u svakom redu na ekranu. Ispisivanje se obavlja pritiskom tastera sa željenim slovima, brojevima i znacima. Pri tome kursor pokazuje na kom mestu ekrana se izvršava ispisivanje. Kursor se može pomerati po ekranu, kao što je rečeno, odgovarajućim komandama. Mogućnost da se tekst može pisati i uređivati po celom ekranu obezbeđena je od strane sistemskog programa u Komodoru koji se naziva ekranski editor (engl. screen editor).

Nakon ispisivanja naredbe ili programske linije pritiskom na taster RETURN ispisano prelazi u memoriju računara. Pri tome je važno sledeće: Komodor će prihvatiti maksimalno do 80 karaktera. To znači da jedna programska linija može biti napisana u dva reda ekrana. Sve ono što je duže od 80 karaktera neće biti prihvaćeno u računar po pritisku tastera RETURN. Za tu osobinu ekranskog editora se kaže da je njegova logička linija dugačka 80 karaktera.

U slučaju da se to pojavi kao ograničenje potrebno je programsku liniju razdvojiti na dve programske linije. Do tako dugih programskih linija moguće je doći korišćenjem dvotačke za razdvajanje naredbi u jednoj programskoj liniji.

Skraćeni način pisanja naredbi

Komodor omogućuje ispisivanje jedne naredbe pritiskom manjeg broja tastera nego što ima karaktera u jednoj naredbi. Time se ubrzava upisivanje programa i podataka u računar.

Skraćeni način ispisivanja se najčešće izvodi pritiskom tastera prvog slova naredbe, a zatim pritiskom tastera drugog slova, ali uz pritisnut taster SHIFT.

Primer: Naredba LIST se skraćeno unosi tako što se pritisne taster L, a zatim uz pritisnut taster SHIFT taster I. Na ekranu će biti ispisano slovo L i grafiki znak koji se nalazi sa desne donje strane tastera I. Pritiskom na taster RETURN program će biti izlistan što potvrđuje da je Komodor prihvatio skraćeno unetu naredbu.

Neke naredbe se nešto drugačije pišu skraćeno. Na primer veoma korišćena naredba za ispisivanje teksta i rezultata na ekranu skraćeno se ispisuje samo jednim karakterom, znakom pitanja (?).

Primer: Naredba:

?5+7

daće rezultat 12.

Spisak svih naredbi I njihovog skraćenog označavanja je dat u dodatku A.

Uzimajući u obzir da ekranski editor omogućuje upisivanje do 80 karaktera po programskoj liniji skraćeni način pisanja naredbi omogućuje upisivanje većeg broja naredbi u jednu programsku liniju.

Izveštaji

Kada prestane da se izvršava bejzik program, računar ispisuje izveštaj kojim objašnjava razlog zaustavljanja. Izveštaj se sastoji od tekstualne poruke i eventualno broja. Tekstualna poruka na engleskom jeziku objašnjava razlog zaustavljanja, a broj ukazuje na broj programske linije u kojoj je prestalo dalje izvršavanje programa.

Prilikom izvršavanja prethodnih primera, kako u direktnom tako i u programskom načinu rada, dobijale su se sledeće poruke: **READY.**, što znači da je program izvršen u potpunosti, **BREAK.**, da je program prekinut, zatim **?SYNTAX ERROR** poruka koja javlja da je napisano nešto što računar ne razume i ne može da izvrši. Poruka **?OUT OF DATA ERROR** je česta poruka u pisanju i proveri rada programa. Javlja se najčešće kada se kursor nalazi na poruci **READY.** i kada se pritisne taster RETURN. U tom slučaju ta poruka nema praktični značaj i jednostavno treba nastaviti sa daljim radom.

Na ovom mestu je izvršeno upoznavanje sa izveštajima, a u delu knjige 4.3 detaljno su objašnjene sve vrste izveštaja.

3.6 UČITAVANJE PROGRAMA

Učitavanje programa je premeštanje programa ili raznih podataka iz spoljne sredine u računar. Programi se najčešće učitavaju sa kasetofona (kasete) ili disk jedinice (diskete).

Učitavanje programa sa kasetofona

U radu sa kasetofonom program se nalazi na magnetofonskoj traci (kaseti). Za učitavanje programa potrebno je spojiti Komodor sa kasetofonom. To se ostvaruje uključenjem kasetofona (koristi se specijalni kasetofon za Komodor tzv. DATA SETTE) na priključak CASSETTE. To je drugi priključak sa desne strane gledano otpozadi. Za vreme priključivanja treba Komodor isključiti iz struje da slučajno ne dođe do električnog oštećenja.

Za učitavanje programa potrebno je zadati odgovarajuću naredbu. To je naredba **LOAD.** Ispisivanjem naredbe i pritiskom tastera RETURN na ekranu će se ispisati poruka **PRESS PLAY ON TAPE.** Ona označava da se pritisne taster PLAY na kasetofonu. Pre toga je potrebno premotati traku ispred početka programa. Naredba **LOAD** se skraćeno može ostvariti jednovremenim pritiskom tastera SHIFT i RUN/STOP.

Na ovaj način se učitava program na koji se naiđe. Nailaskom na program računar će javiti da je našao program: FOUND "ime programa". Pritiskom na taster praznog polja (razmaknicu) počinje učitavanje programa. Trajanje učitavanja zavisi od dužine programa i za duže programe može trajati više minuta.

Ako je učitavanje uspelo računar prijavljuje izveštaj **READY.** i zaustavlja kasetofon. Nakon toga se može izvršiti naredba **RUN** i startovati program.

Detaljnije o učitavanjima i snimanjima programa je dato u delu knjige 4.2 pod naslovom "Rad sa kasetofonom".

Učitavanje programa sa disk jedinice

Disk jedinica je uređaj koji omogućuje brže nalaženje i učitavanje programa. On koristi magnetne diskete za snimke programa i podataka. Priključuje se u treći priključak (SERIAL) sa desne strane gledano otpozadi. Prilikom priključivanja preporuka je da je i Komodoru i disku prekinuto napajanje iz mreže.

Naredba za učitavanje sa diska je **LOAD"*",8.** Ovom naredbom će se učitati prvi program sa diska. Ako se umesto zvezdice navede ime programa biće učitan program sa tim imenom.

Jedan broj programa zahteva za učitavanje nešto izmenjenu naredbu. To je naredba **LOAD "ime program", 8,1.**

Nakon učitanog programa može se pristupiti njegovom startovanju naredbom **RUN** ili nekom drugom koja je navedena u uputstvima za korišćenje programa.

Detaljnije o radu sa disk jedinicom je dato u delu knjige 4.2 pod naslovom "Rad sa disk jedinicom".

4
Bejzik

Bejzik (BASIC) je, uz paskal (PASCAL) i fortran (FORTRAN), danas jedan od najčešće korišćenih viših programskih jezika. Razvijen je početkom šezdesetih godina na Dartmaut (Dartmouth) univerzitetu kao pomoćno sredstvo za učenje programiranja. Prvi put je primenjen na računaru 1965. godine. Iako je od tada pretrpeo veliki broj izmena, što je dovelo do nestandardizacije i pojave različitih varijanti bejzika, zadržao je svoje osnovne prednosti nad drugim programskim jezicima. Te prednosti su:

— Lak je za učenje i upotrebu. Broj formalnosti je manji nego kod većine drugih programskih jezika. U njemu se koriste osnovne reči engleskog jezika. Interpreterski način rada olakšava prve korake u njegovom korišćenju.

— Veoma rasprostranjen programski jezik. Na mikroračunarima (kućni i lični računari) skoro je uvek korišćen. Postoji njegova velika podrška u brojnim knjigama i časopisima koji su posvećeni računarima.

Glavni nedostaci bejzik programskog jezika su, uz spomenutu nestandardizaciju: sporo izvršavanje programa, što važi i za kompilirane verzije, neekonomično trošenje memorije i često nepodržavanje struktuiranog programiranja.

Postoji veći broj programskih jezika koji nemaju nedostatke bejzika i koji su moćniji od njega, ali je za njihovu upotrebu potrebna veća obučenost. Takođe, većina ih je specijalizovana za određenu vrstu primene. Sve to čini da je bejzik programski jezik pronašao primenu u najrazličitijim oblastima. Od igara, preko učenja programiranja, do poslovne i naučne primene.

U ovom delu knjige prikazaće se bejzik Komodora 64. Za razumevanje izloženog podrazumeva se da je čitalac pažljivo proučio prethodna poglavlja. Ovo poglavlje je sigurno najznačajnije za korišćenje i razumevanje rada Komodora, i saglasno tome zahteva odgovarajuću pažnju. Kao logični i neophodni nastavak na ovo poglavlje nadovezuje se naredno poglavlje o principima programiranja, čime se obuhvata sve što je neophodno za programiranje Komodora u bejziku. Time se otvara put ka širim mogućnostima programiranja i korišćenja Komodora (Sajmons bejzik, mašinsko programiranje...) izloženih u ostalim poglavljima.

4. 1 OSNOVNI POJMOVI

Karakteri

Karakteri su znakovi koje Komodor koristi za rad u bejziku. To su: slova (mala i velika), cifre, matematički i grafički simboli, znaci interpunkcije i kontrolni znaci načina ispisivanja.

Svaki karakter je označen brojem koji se naziva kôd karaktera ili samo kôd.

Brojevi

Za rad u bejziku na Komodoru koriste se decimalni brojevi. Pri tome je značajno sledeće:

- umesto kod nas uobičajenog decimalnog zareza koristi se decimalna tačka,
- nula se predstavlja ovim znakom: Ø,
- prazna mesta između cifara su dozvoljena,
- broj može početi decimalnom tačkom.
- predznak (+ ili −) navodi se ispred broja, a ako nije naveden, broj je pozitivan,
- broj se može predstaviti i u eksponencijalnom obliku.

Primeri:

pravilno	nepravilno
3.14	3,14
23400	23,400
23 400	1.000.000
0.91	
.91	
234E2	

U eksponencijalnom obliku broj je sastavljen od tri dela:

- mantise
- slova E
- eksponenta

Slovo E ukazuje da je brojna vrednost u eksponencijalnom obliku. Eksponent se navodi iza slova E i može biti samo ceo broj. Ukazuje sa kojim stepenom broja 10 treba pomnožiti mantisu da bi se dobila brojna vrednost broja datog u eksponencijalnom obliku (u gornjem primeru $234E2 = 234 \times 10^2 = 23400$).

I mantisa i eksponent mogu imati predznak. Predznak mantise određuje da li je broj pozitivan ili negativan. Predznak eksponenta određuje na koju stranu je potrebno pomeriti decimalnu tačku mantise da bi se broj preveo iz eksponencijalnog u normalni oblik.

Primeri:

12.8E2	=1280
−1.28E2	=−128
12.8E−2	=.128
−12.8E−11	=−.000000000128

Opseg brojeva sa kojima Komodor može da radi je od 2.93873588E−39 do 1.70141183E38 kako pozitivnih tako i negativnih. Njihova tačnost može biti 9 ili 10 cifara, ali se prilikom prikazivanja zaokružuju na tačnost od 9 cifara.

U slučaju pozitivnog broja većeg od najvećeg mogućeg ili negativnog manjeg od najmanjeg mogućeg prijaviće se izveštaj o grešci **OVERFLOW ERROR**. Pozitivni brojevi manji od najmanjeg mogućeg i negativni veći od najvećeg postaju nula i pri tome se ne javlja izveštaj o grešci.

Brojevi manji od .01 i veći od 999999999 prikazuju se u eksponencijalnom obliku.

Brojne promenljive

Brojna promenljiva je simbolička oznaka kojoj se može dodeliti brojna vrednost.

Primer: A=3.14

Brojna promenljiva ima ime A i tekuću vrednost 3.14.

Dodeljivanje vrednosti promenljivoj ostvaruje sa odgovarajućim bejzik naredbama (**LET, INPUT, READ**...). U slučaju da se upotrebljava promenljiva kojoj prethodno nije dodeljena vrednost, njena vrednost će biti nula.

Ime brojne promenljive može biti proizvoljno dugačka kombinacija slova i cifara, ali samo prva dva karaktera su od značaja. To znači da se bar jedan od prva dva karaktera imena jedne promenljive mora razlikovati od odgovarajućeg karaktera imena druge promenljive.

Ime mora počinjati slovom. U imenu se mogu sadržati karakteri praznog mesta (engl. space) i kontrolni karakteri boje, ali ga ne menjaju.

Imena promenljivih ne smeju biti reči rezervisane za bejzik (imena naredbi, funkcija). Takođe se unutar imena promenljive ne sme nalaziti kombinacija slova koja predstavlja rezervisanu reč.

Po načinu na koji se brojna vrednost predstavlja i obrađuje u računaru razlikuju se dve vrste brojnih promenljivih:

1. Celobrojne promenljive
2. Realne promenljive

Vrednost celobrojne promenljive mora biti ceo broj iz opsega od − 32768 do +32767.

Primeri:

ispravno	neispravno	
1985	1985.2	— nije ceo broj
32768	32769	— izvan opsega
−202	−.202	— nije ceo broj

Ceo broj predstavljen u memoriji računara zauzima dva bajta.

Vrednosti realnih promenljivih pokrivaju ranije navedeni ceo opseg brojeva sa kojima Komodor može da radi. Realni brojevi sa kojima računar radi nazivaju se brojevi sa pomičnim zarezom (engl. floating point).

Broj sa pomičnim zarezom u memoriji računara zauzima pet bajtova.

U imenu celobrojne promenljive poslednji karakter mora da bude znak za procenat (%).

Primeri:

A=231.78	realna promenljiva
TCR=35E5	realna promenljiva
SUMA=.0077	realna promenljiva
JK%=28	celobrojna promenljiva
TDD%=−5409	celobrojna promenljiva

Brojni izrazi

Brojni izraz određuje postupak za izračunavanje brojne vrednosti. Sastoji se iz jednog ili više argumenata (operanada) nad kojima se izvršavaju odgovarajuće operacije.

Primer: $\sqrt{4}+3$ (u bejziku **SQR(4)+3**)

Za određivanje vrednosti navedenog izraza potrebno je izvršiti operaciju korenovanja argumenata 4, a zatim rezultat (argument 2) sabrati sa argumentom 3.

Argumenti mogu biti brojne vrednosti (konstante) ili brojne promenljive. Operacije koje se mogu izvoditi nad argumentima su: aritmetičke, trigonometrijske, eksponencijalne, relacione i logičke. Njihov rezultat mora biti brojna vrednost.

Stringovi

String je niz karaktera (kao što je broj niz cifara). Ti karakteri mogu biti slova, cifre i ostale vrste znakova sa kojima Komodor raspolaže. Takođe se mogu koristiti karakter praznog polja, kontrolni karakteri boje i načina ispisivanja. Ako string ne sadrži ni jedan karakter, on se naziva prazan string.

Jedini karakter koji se ne može koristiti unutar stringa je navodnik ("). Razlog za to je što se on koristi za označavanje početka i kraja stringa.

Dužina stringa je kod Komodora ograničena na 255 karaktera.

Primeri: **"AVION"**
"COMMODORE 64"
"* #KKK$$%+"
" "
"23.4"
"" — prazan string

String promenljive

Slično brojnoj promenljivoj, string promenljiva je simbolička oznaka kojoj se može dodeliti string vrednost.

Primer: **A$="HL6"**

Ime string promenljive je A$, a tekuća string vrednost je HL6.

Dodeljivanje vrednosti string promenljivama se obavlja kao i brojnim promenljivima. String promenljiva kojoj nije prethodno dodeljena vrednost imaće vrednost praznog stringa.

Za ime string promenljive važi isto što i za ime brojne promenljive, s tim što poslednji karakter u imenu mora biti znak za dolar ($).

String izrazi

Izraz sa stringovima (string izraz) određuje postupak za nalaženje rezultujuće vrednosti, brojne ili string. Sastoji se iz jednog ili više argumenata, od kojih je bar jedan string tipa, nad kojima se izvršavaju odgovarajuće operacije. Argumenti mogu biti brojne vrednosti, stringovi, brojne i string promenljive.

4.2 NAREDBE I NJIHOVA UPOTREBA

U ovom delu knjige su opisane Komodorove bejzik naredbe i načini njihovog korišćenja. Naredbe su date po redosledu koji omogućava njihovo postepeno upoznavanje, od najčešće korišćenih do naredbi posebne namene. Pri tome je izvršena njihova podela u logičke celine, zavisno od namene. S obzirom da naredbe nisu izložene po abecednom redu, za brzo nalaženje određene naredbe treba koristiti indeks dat na kraju knjige.

Komodor raspolaže sa više od 60 nezavisnih naredbi. Pri tome se za njihovo opisivanje koristi 71 rezervisana reč (engl. keywords). Rezervisane reči su izvedene iz reči engleskog jezika na osnovu namene naredbe.

U okviru svih naredbi razlikuju se naredbe čiji je rezultat brojna ili string vrednost. One se nazivaju funkcijske naredbe ili samo funkcije.

Naredbe se mogu unositi u računar i u skraćenom obliku. Pri tome nije potrebno upisati sve karaktere naredbe, već najčešće odgovarajuća dva ili tri karaktera. Po upisivanju prvog karaktera (i drugog) naredni se upisuje uz pritisnuti taster SHIFT. Time je moguće

delimično prevazići ograničenje Komodorovog ekranskog editora koji omogućava upisivanje do 80 karaktera po programskoj liniji. Svaka rezervisana reč se u memoriji računara pamti kao jedan bajt, tj. kao jedan karakter (engl. token), tako da skraćeni način upisivanja nema uticaja na potrošnju memorije.

U prvoj koloni tabele 1 (Dodatak) dat je spisak rezervisanih reči Komodorovog bejzika. U drugoj koloni je dat skraćeni način njihovog upisivanja u računar. U trećoj su dati karakteri koji se ispisuju na ekranu pri skraćenom načinu upisivanja naredbi.

4.2.1 Osnovne naredbe

NEW

Pre unošenja novog programa u računar treba pomoću naredbe **NEW** iz RAM memorije izbrisati prethodni program i vrednosti njegovih programskih promenljivih.

Izvršenjem naredbe **NEW** bejzik program praktično se ne briše, već se samo odgovarajuće sistemske promenljive postavljaju na početne vrednosti.

Potpuno brisanje RAM-a postiže se isključivanjem računara ili izvršavanjem naredbe **SYS 64738.**

RUN

Kada je program unet u memoriju, njegovo izvršavanje otpočinje naredbom **RUN.** Može se zadati direktno ili ,ređe, u samom programu.

Naredbom **RUN** izvršavanje programa će otpočeti od programske linije sa najmanjim brojem. Stavljanjem celog broja iza naredbe **RUN,** izvršavanje programa otpočinje od programske linije sa tim brojem. Ako ta programska linija ne postoji, pojaviće se izveštaj o grešci **UNDEF'D STATEMENT.**

Broj linije se može zadati i u obliku promenljive ili izraza. Ako njihova vrednost nije ceo broj, biće zaokružena odsecanjem (na manju celobrojnu vrednost).

Izvršavanjem naredbe **RUN** n (n je broj programske iinije) brišu se sve prethodne programske promenljive i njen efekat je isti kao u slučaju naredbe **CLR**: **GO TO n.**

Primeri:	**RUN**	– program se izvršava od programske linije sa najmanjim brojem
	RUN 40	– program se izvršava od linije 40
	RUN 3.7	– program se izvršava od linije 3

LIST

Naredba **LIST** omogućava prikazivanje teksta (listinga) prethodno unetog bejzik programa. Njegovo prikazivanje će se obaviti na ekranu, što dozvoljava upotrebu ekranskog editora za unošenje novih programskih linija i za menjanje postojećih.

Osim za prikazivanje programa na ekranu, naredba može biti upotrebljena i u radu sa štampačem, disk jedinicom ili nekim drugim spoljnim uređajem, što je objašnjeno u naredbi **CMD.**

Ako se iza naredbe **LIST** na navede ništa, biće prikazan ceo program. Za prikazivanje jedne programske linije iza **LIST** treba navesti broj te linije. Ako se iza broja stavi znak – program će biti prikazan od navedene programske linije do krajnje. Stavljanjem znaka –

ispred broja program će biti prikazan od prve programske linije do navedene. Za prikazivanje grupe linija iza naredbe treba navesti brojeve prve i poslednje linije u grupi, sa znakom – između njih.

Primeri:	**LIST**	– prikazuje se ceo program
	LIST 20	– prikazuje se samo linija 20
	LIST 20-	– prikazuje se program od linije 20 do kraja
	LIST –20	– prikazuje se program od početka do linije 20
	LIST 20–100	– prikazuju se programske linije od linije 20 do linije 100

Brojne vrednosti navedene u **LIST** naredbi ukoliko nisu celobrojne zaokružuju se odsecanjem, a mogu biti zadate i u obliku brojnih promenljivih ili izraza.

Za vreme prikazivanja teksta programa (listanja), vrši se njegovo pomeranje po ekranu odozdo nagore i ono se može usporiti pritiskom na taster CTRL, ili zaustaviti pritiskom tastera RUN/STOP.

Naredba **LIST** se po pravilu zadaje direktno, mada je moguće i njeno unošenje u program. To će dovesti do prikazivanja i zaustavljanja programa te ispisivanja izveštaja **READY.**

STOP

Naredbom **STOP** obustavlja se dalje izvršavanje bejzik naredbi.

Naredba može biti zadata programski, a retko i direktno. U prvom slučaju ispisuje se izveštaj **?BREAK IN** n, gde je n broj linije u kojoj se nalazi naredba **STOP** koja je prekinula izvršavanje programa. U drugom slučaju prijavljuje se izveštaj **BREAK.**

Nastavljanje izvršavanja zaustavljenog programa može se ostvariti naredbama **CONT** ili **GOTO.**

Primer:
```
10 FOR N=1 TO 10:PRINT N:NEXT
20 STOP
30 FOR N=11 TO 20:PRINT N:NEXT
```

Nakon ispisivanja brojeva od 1 do 10 program se zaustavlja i ispisuje se izveštaj **BREAK IN 20.** Direktnim zadavanjem naredbe **CONT** prelazi se na izvršavanje linije 30 u kojoj se ispisuju brojevi od 11 do 20.

STOP naredba ima isti učinak kao pritisak na taster **RUN/STOP.**

END

Naredba **END** okončava izvršavanje programa. Pri tom ispisuje poruku **READY** i predaje korisniku upravljanje računarom.

Može se postaviti bilo gde u programu. Razlikuje se od **STOP** naredbe u tome što ne ispisuje poruku o prekidu **BREAK IN** n.

Primer: **120 IF N>100 THEN END**

Linijom 120 iz nekog zamišljenog programa završava se njegovo izvršavanje ako je promenljiva N veća od 100. U protivnom izvršavanje se nastavlja od sledeće linije.

Izvršenje programa se može nastaviti naredbom **CONT.**

CONT

Naredbom **CONT** nastavlja se izvršavanje prekinutog programa.

Može se upotrebiti nakon pritiska tastera RUN/STOP ili posle izvršenja naredbi **STOP** ili **END.** Program se nastavlja od mesta prekida.

CONT se uobičajeno koristi uz naredbu **STOP** u toku razvijanja programa. Kada je program zaustavljen, može se pristupiti proveri vrednosti promenljivih. Mogu im se menjati vrednosti direktnim načinom rada. Nakon toga, direktnim zadavanjem naredbe **CONT** nastavlja se izvršavanje programa.

Izveštaj o grešci **CAN'T CONTINUE** (ne može se nastaviti) pojaviće se ako se pristupilo izmeni (editovanju) programa, ili ako je naredbi **CONT** prethodio izveštaj o grešci.

Primer:
```
10 FOR A= 1 TO 200
20 PRINT A;
30 NEXT A
```

U toku izvršavanja programa, dok traje ispisivanje brojeva, treba pritisnuti taster RUN/STOP. Nakon ispisivanja izveštaja o prekidu program se nastavlja direktnim zadavanjem naredbe **CONT.**

REM

U cilju bolje preglednosti programa, naročito u toku njegovog razvijanja, potrebno je označiti ili objasniti pojedina mesta u programu. To se može ostvariti naredbom **REM** (od engl. remark — primedba). Iza nje se može ispisati proizvoljan tekst uz upotrebu svih karaktera. Takođe se mogu navoditi i naredbe razdvojene dvotačkom, pri čemu se one neće izvršavati. Jedino CHR(13) (taster RETURN) ima dejstvo nakon naredbe **REM.**

Primer:
```
100 REM NALAZENJE KVADRATA PRVIH 10 BROJEVA
110 FOR N=1 TO 10
120 PRINT N,N*N:REM ISPISIVANJE
130 NEXT
```

REM naredbe usporavaju izvršavanje programa i povećavaju utrošak memorije.

PRINT

Ovom naredbom se na ekranu ispisuju rezultati izračunavanja, tekst ili različiti karakteri. Takođe se može upotrebiti, uz pomoć naredbe **CMD,** za njihovo slanje na spoljnje jedinice (štampač, disk jedinicu, kasetofon, modem).

Načini upotrebe naredbe **PRINT**:

1. Ispisivanje brojeva, vrednosti brojnih promenljivih i vrednosti izraza.

	naredba	ekran
Primer:	**PRINT 2+3**	**5**

Primer: **10 FOR N = −10 TO 11:PRINT N,9 ↑N:NEXT**

Ovim primerom ispisuju se vrednosti promenljive N i broja 9 dignutog na N-ti stepen, za promenu N od −10 do +11.

Prilikom ispisivanja brojeva Komodor iza svakog broja ispisuje i jedan prazan karakter (prazno polje), a ako je broj pozitivan onda i ispred njega.

2. Ispisivanje stringova, vrednosti string promenljivih i izraza sa stringovima. Ispred i iza stringa obavezni su navodnici.

Primer:	naredba	ekran
	PRINT"1985.GOD."	**1985.GOD.**

Primer:	**10 LET A$="COMMODORE"**	
	20 PRINT A$	**COMMODORE**

Unutar navodnika mogu se upisivati kontrolni karakteri. Oni će imati uticaja na način ispisivanja (mesto, boju...). Razlikuju se sledeći karakteri i kontrolni načini koji se mogu navesti unutar navodnika.

a. Komande za pomeranje kursora

Pritiskanjem tastera za pomeranje kursora, unutar navodnika, utiče se na mesto ispisivanja. Za svaki pritisak tastera Komodor će unutar navodnika ispisati sledeće grafičke karaktere:

pritisnut taster	na ekranu
CLR/HOME	S
SHIFT CLR/HOME	♥
⇑ CRSR ⇓	Q
SHIFT ⇑ CRSR ⇓	●
⇐CRSR⇒	]
SHIFT ⇐CRSR⇒	▌

Primer: **PRINT"♥"** pritisnut taster SHIFT CLR/HOME

Briše se sadržaj ekrana, isto kao da je SHIFT CLR/HOME pritisnut direktno. Na taj način se brisanje ekrana ostvaruje programski. Isto tako u program se mogu uneti i ostale komande i time postići njihova dejstva.

U sledećim primerima nisu dati grafički simboli već potrebni tasteri za njihovo dobijanje.

Primer:
```
10 PRINT"{CLR}"
20 PRINT"I{C/DN}S{C/DN}T{C/DN}I{C/DN}PRI{C/RT}M{C/RT}E{C/RT}R"
```

Unošenjem komandi za pomeranje kursora nadole, i nadesno, reč ISTI se ispisuje dijagonalno, a drugi deo reči PRIMER horizontalno sa po jednim praznim poljem između slova.

Od komandi za uređivanje teksta, tj. kursorskih komandi, svoje uobičajno dejstvo unutar navodnika zadržava samo INST/DEL komanda.

b. Inverzni karakteri

Komodor može karaktere ispisivati i inverzno (kao negativ fotografije). Prelazak na inverzni način (engl. reverse video) ispisivanja ostvaruje se istovremenim pritiskanjem tastera CTRL i 9. Tada se na ekranu, unutar navodnika, pojavljuje znak R, ukazujući da će naredni karakteri, izvršenjem naredbe, biti inverzni.

Povratak na normalno ispisivanje ostvaruje se pritiskanjem tastera CTRL i 0 (pri tome se pojavljuje znak ⊟).

Primer: `10 PRINT"{RVON}MIKRO{RVOF}KNJIGA"`

Izvršenjem naredbe reč MIKRO ispisuje se inverzno.

c. Kontrole boje

Karakteri se mogu ispisivati u jednoj od 16 boja. Za to je potrebno pre unošenja karaktera, unutar navodnika, pritisnuti odgovarajuće tastere.

Boje, tasteri i znaci koji se pri tome pojavljuju su sledeći:

taster	boja	na ekranu
CTRL 1	crna	
CTRL 2	bela	E
CTRL 3	crvena	£
CTRL 4	cijan	
CTRL 5	purpurna	
CTRL 6	zelena	
CTRL 7	plava	
CTRL 8	žuta	π
C= 1	narandžasta	
C= 2	smeđa	
C= 3	svetlocrvena	
C= 4	siva 1	
C= 5	siva 2	
C= 6	svetlozelena	
C= 7	svetloplava	
C= 8	siva 3	

Primer: `10 PRINT "{GRN}MIKRO{BLUE}K{YELO}NJ{ORNG}I{BRN}G{LRED}A"`

Reč MIKRO se ispisuje u zelenoj boji, slovo K u plavoj, NJ u žutoj, I u narandžastoj, G u smeđoj i A u svetlocrvenoj boji.

d. Dopisivanje i brisanje

Dopisivanje i brisanje (taster INST/DEL) zadržava svoj normalni način rada i unutar navodnika.

e. Ostale kontrole

Karakteri koji omogućavaju preostale kontrole unose se između navodnika po nešto složenijem postupku. Potrebno je ostaviti prazno mesto unutar navodnika, pritisnuti taster RETURN i vratiti se, upotrebom kursorskih komandi, na prazno mesto. Tada treba preći u inverzni način ispisivanja (CTRL 9), i pritisnuti naznačene tastere za postizanje željenih funkcija.

funkcija	pritisnut taster	znak
prelazak na mala slova	N	N
prelazak na velika slova	SHIFT N	/
zabrana promene slova	H	H
dozvola promene slova	I	I
SHIFT RETURN	SHIFT M	\

3. Znakovi interpunkcije omogućavaju ispisivanje na određenom mestu onoga što je navedeno iza njih.

– tačka zarez ; odmah iza znaka
– zarez , od sledeće četvrtine reda

Primeri:

naredba	ekran
PRINT 1:2:3:4	1 2 3 4
PRINT 1,2	1 2

U prvom od prethodna dva primera dva prazna polja između brojeva odgovaraju praznim karakterima koji se ispisuju ispred i iza pozitivnih brojeva. Devet praznih polja, u drugom primeru, odgovara jednoj četvrtini jednog reda ekrana.

Na ekranu se može ispisati ukupno 1000 karaktera, u 25 redova sa po 40 karaktera u svakom redu (25 redova i 40 kolona). Svaki red je podeljen u 4 dela od po 10 karaktera. Zarez (,) prenosi ispisivanje u sledeću četvrtinu reda, a to može biti i u sledećem redu.

U slučaju stringova tačka zarez (;) može se izostaviti.

4. Ako se iza naredbe PRINT ne navede ništa, pri izvršavanju sledeće PRINT naredbe ispisivanje će otpočeti jedan red niže.

Primer:

naredba	ekran
PRINT 1:PRINT:PRINT 2	**1**
	2

5. Funkcija **TAB** omogućava ispisivanje u željenoj koloni. Naredba tada ima oblik:

PRINT TAB (X) ono što se ispisuje

a ispisivanje počinje od mesta koje je udaljeno X karaktera od početka reda. Argument X može imati vrednost od 0 do 255, može biti dat i u obliku promenljive ili izraza, a ako nije celobrojan zaokružuje se na manji ceo broj. U slučaju neodgovarajuće vrednosti argumenta pojaviće se izveštaj o grešci **ILLEGAL QUANTITY.**

Između reči TAB i zagrade ne sme postojati prazno polje.

Primer:
```
10 FOR N=0 TO 10
20 PRINT TAB(N)N
30 NEXT
```

Brojevi od 0 do 10 ispisuju se dijagonalno.

6. Funkcija **SPC** određuje koliko će se praznih polja ispisati do sledeće PRINT pozicije. Naredba treba da ima oblik:

PRINT SPC(X) ono što se ispisuje.

Za razliku od **TAB, SPC** određuje broj praznih mesta u odnosu na poslednje mesto ispisivanja. Za argument X važi navedeno pod TAB. U slučaju primene naredbe SPC na disk jedinicu, X može biti od 0 do 254.

Primer:
```
10 PRINT TAB(12)"LOGICKA STANJA"
20 PRINT "ULAZ A"SPC(10)"ULAZ B"SPC(13)"IZLAZ
```

Na ovaj način se formira zaglavlje tabele.

POS(N)

Ova naredba, tj. funkcija, omogućuje utvrđivanje kolone u kojoj je izvršeno poslednje ispisivanje karaktera. Njen rezultat može biti samo ceo broj od 1 do 40. Broj 1 odgovara krajnje levoj koloni, a 40 krajnje desnoj.

Broj N koji se navodi u zagradi može biti bilo koji, ali je obavezan.

Primer:
```
10 FOR X=0 TO 40
20 PRINT TAB(X)"*";POS(0)
30 NEXT X
```

Znak * ispisuje se redom u svakoj koloni. Takođe se ispisuje i broj kolone nađen naredbom POS. U toku izvršavanja program se može usporiti pritiskom na taster CTRL.

LET

Naredba **LET** služi za dodeljivanje vrednosti promenljivama. To se može uraditi još i pomoću naredbi **INPUT** i **READ.** Promenljiva zadržava dodeljenu vrednost sve dok joj se ne dodeli nova.

– brojne promenljive

Dodeljivanje vrednosti brojnoj promenljivoj ostvaruje se naredbom u obliku:

LET a = b

gde je **a** ime promenljive, a **b** vrednost koja joj se dodeljuje. **b** može biti broj, brojna promenljiva ili izraz. Reč **LET** nije obavezna, što omogućava dodeljivanje vrednosti u obliku:

a = b

čime se dobijaju programi koji su kraći i koji se brže izvršavaju.

Primer:
```
10 LET A=4
20 AMIGA=A
30 CX32=A+SQR(AMIGA)
40 PRINT A,AMIGA,CX32
```

U liniji 10 promenljiva A dobija vrednost 4, u liniji 20 promenljiva AMIGA dobija vrednost promenljive A, u liniji 30 promenljiva CX32 dobija vrednost zbira vrednosti promenljive A i kvadratnog korena promenljive AMIGA, a naredbom u liniji 40 ispisuju se vrednosti sve tri promenljive.

— string promenljive
Dodeljivanje vrednosti string promenljivoj ostvaruje se na isti način kao i brojnoj promenljivoj. String promenljivoj se može dodeliti vrednost stringa, string promenljive ili izraza sa stringovima.

Primer:
```
10 A$="MIKRO"
20 B$=A$
30 A$=B$+" KNJIGA"
40 PRINT A$
```

U liniji 10 se string promenljivoj A$ dodeljuje string MIKRO. Ispred i iza stringa su obavezni znaci navoda. U liniji 20 promenljivoj B$ dodeljuje se vrednost promenljive A$, a u liniji 30 promenljiva A$ dobija vrednost zbira vrednosti promenljive B$ i stringa KNJIGA (uočiti prazno polje ispred slova K). Naredbom u liniji 40 ispisuje se vrednost promenljive A$.

Dodeljivanje vrednosti višedimenzionalnim promenljivama je prikazano u opisu naredbe **DIM**.

CLR

Ova naredba svim brojnim promenljivama u programu dodeljuje vrednost nula, a svim string promenljivama prazan string. Pri tome se sam bejzik program, koji se nalazi u memoriji, ne briše.

INPUT

Pomoću naredbe **INPUT** korisnik dodeljuje vrednosti promenljivama u toku izvršavanja programa, direktno preko tastature.

Izvršavajući ovu naredbu računar čeka na unošenje vrednosti. Pri tom ispisuje znak pitanja (**?**) na ekranu. Sa desne strane znaka ostavlja jedno prazno polje, a na njega postavlja kursor. Unošenje podataka se završava pritiskom na taster RETURN.

Primer:
```
10 INPUT X
20 PRINT X "NA KVADRAT JE" X^2
```

Izvršavajući naredbu u liniji 10 računar očekuje unošenje brojne vrednosti namenjenu promenljivoj X. Upisivanjem broja i pritiskom na taster RETURN prelazi se na liniju 20. Ispisuje se uneseni broj, poruka i izračunata vrednost njegovog kvadrata.

Ako se ne upiše broj pojaviće se poruka **?REDO FROM START** označavajući da je unesen string, a ne broj. Ako se na pojavu znaka pitanja ne unese vrednost, već samo pritisne taster RETURN, promenljiva će zadržati svoju vrednost.

U naredbu **INPUT** može se uključiti i poruka. Ona će se pojaviti ispred znaka pitanja objašnjavajući šta je potrebno upisati. Poruku, koja može biti bilo kakav tekst, treba navesti unutar, navodnika iza reči **INPUT.** Takođe je potrebno odvojiti je tačkom i zarezom (;) od promenljive.

Primer: Liniju 10 u gornjem primeru treba zameniti sa:
```
10 INPUT "UPISITE BROJ";X
```

Jednom **INPUT** naredbom mogu se dodeljivati vrednosti većem broju promenljivih. U tom slučaju potrebno je iza reči INPUT navesti imena promenljivih, međusobno odvojena zarezima.

Primer:
```
10 INPUT A,B,C
20 PRINT A:PRINT B:PRINT C
```

Izvršavanjem linije 10 pojavljuje se jedan znak pitanja. Unošenjem brojne vrednosti i pritiskom na taster RETURN pojavljuju se dva znaka pitanja **(??)**, označavajući da je potrebno dalje unošenje vrednosti. Vrednosti se mogu uneti i međusobno odvojene zarezima. Time se izbegava pojava dva znaka pitanja za svaku novu vrednost. Ako se unese više vrednosti nego što je navedeno promenljivih, pojaviće se poruka **?EXTRA IGNORED,** označavajući da višak unesenih vrednosti nije uzet u obzir.

Dodeljivanje vrednosti string promenljivama obavlja se na isti način kao i brojnim, s tim što je potrebno navesti ime string promenljive, a pri dodeljivanju vrednosti uneti string.

Primer:
```
10 INPUT "MOJE IME JE";A$
20 FOR N=0 TO 10
30 PRINT TAB(N)A$
40 NEXT
```

String koji se unosi, kao odgovor na **INPUT** naredbu, ne mora imati navodnike na početku i kraju, osim ako ne sadrži zareze ili prazna polja na početku ili kraju.

INPUT naredba ne može se primenjivati direktno, već samo programski.

GET

Naredbom **GET** očitava se pritisnuti taster.

Rezultat se dodeljuje promenljivoj koja se navodi iza reči **GET.** Promenljiva može biti brojna ili string. Ako je navedena brojna promenljiva i ako se pritisne taster koji nije broj, prijaviće se izveštaj o grešci **?SYNTAX ERROR.** Zbog toga se preporučuje dodeljivanje vrednosti string promenljivoj i njeno prevođenje u brojnu vrednost.

Primer: Sledeći primer omogućava ispisivanje teksta kao pisaćom mašinom.

```
10 GET A$:IF A$<>""THEN GOTO 10
20 GET A$:IF A$="" THEN GOTO 20
30 PRINT A$;:GOTO 10
```

U liniji 10 nalaže se računaru da izvršava liniju 10 sve dok se pritisnuti taster ne pusti. Linija 20 se izvršava sve dok se ne pritisne neki taster. U liniji 30 ispisuje se karakter pritisnutog tastera i odlazi na ponovno očitavanje tastature.

GET naredba omogućava očitavanje do 10 pritisnutih tastera. Tada je potrebno imena promenljivih, kojima se dodeljuju karakteri pritisnutih tastera, navesti iza reči **GET** i razdvojiti zarezima.

Primer: **120 GET A$,B$,A,B**

Linijom 120 iz nekog zamišljenog programa očitavaju se četiri tastera od kojih poslednja dva moraju biti tasteri cifara.

GET se može koristiti samo programski. U direktnom načinu rada prijaviće se izveštaj o grešci **?ILLEGAL DIRECT ERROR.**

READ

Naredba **READ** očitava (engl. read) vrednosti navedene u **DATA** naredbi i dodeljuje ih promenljivama (videti **DATA** naredbu).

Promenljive, brojne ili string, navode se iza **READ** naredbe međusobno odvojene zarezima. Vrednosti koje im se dodeljuju navedene su iza **DATA** naredbe takođe odvojene zarezima.

Primer:

```
10 PRINT "BROJ","IME",,"TELEFON"
20 READ A,B$,C
30 PRINT A,B$,,C
40 DATA 1,ACIC ACA,67890
```

Ovaj program očitava brojne i string vrednosti iz **DATA** naredbe u liniji 40. String ACIC ACA ne zahteva navodnike na svom kraju i početku jer u sebi ne sadrži zareze ili prazna polja na početku ili kraju.

Naredba **READ** uvek se koristi uz naredbu **DATA.** Dodeljivanje vrednosti se obavlja tako što se prvoj promenljivoj iz **READ** naredbe dodeljuje prva vrednost iz **DATA** naredbe, drugoj promenljivoj druga vrednost i tako redom. Ako se brojnoj promenljivoj dodeljuje string vrednost, prijaviće se izveštaj o grešci **?SYNTAX ERROR.** Pri tome će se u izveštaju o grešci prijaviti broj linije u kojoj se nalazi naredba **DATA.** Ako je broj promenljivih veći od broja navedenih vrednosti, prijaviće se izveštaj o grešci **?OUT OF DATA.**

Jedna **READ** naredba može dodeljivati vrednosti promenljivama pomoću jedne ili više DATA naredbi.

Primer:

```
10 READ A,B$,C$
20 PRINT A,B$,C$
30 DATA 1,DA
40 DATA NE
```

Više od jedne READ naredbe može se upotrebiti za dodeljivanje vrednosti iz iste DATA naredbe.

Primer:

```
10 READ A,B$
20 READ C
30 PRINT A,B$,C
40 DATA 1,KNJIGA,2
```

Primer: Prikazan je jedan način dodeljivanja vrednosti elementima matrice.

```
10 DIM A(2,3)
20 FOR N=0 TO 2:FOR M=0 TO 3
30 READ A(N,M)
40 PRINT A(N,M),
50 NEXT M:PRINT
60 NEXT N
70 DATA 15,6,34,0,81,4,-8,6,3,43,36,7
```

DATA

Naredba **DATA** služi za smeštanje brojnih i string vrednosti koje se očitavaju pomoću naredbe READ.

Vrednosti se navode iza reči DATA i međusobno su odvojene zarezima. Ne mogu se navesti u obliku promenljivih ili izraza. Brojne promenljive mogu biti date u bilo kom obliku. Kao celobrojne ili sa pomičnim zarezom, a time i u eksponencijalnom obliku. String vrednosti u **DATA** naredbi ne moraju biti unutar navodnika, osim ako u stringu nisu uključeni: zarez (,), dvotačka (:) prazna polja, grafički ili kontrolni karakteri.

DATA naredba nije izvršna. To znači da se nailaskom na nju, pri izvršavanju programa, odmah prelazi na sledeću naredbu. To omogućava da se **DATA** naredbe postave bilo gde u programu. Najčešće se stavljaju na kraj programa. Time se dobija program koji je pregledniji i koji se brže izvršava.

Sve vrednosti navedene iza reči **DATA** čine niz koji se naziva datoteka. Nju čine sve vrednosti navedene u više **DATA** naredbi. Bez obzira koliko ima vrednosti po naredbi ili gde je ona u programu, READ naredba očitava po redu vrednosti u datoteci.

Primer: Videti primere za READ naredbu.

RESTORE

RESTORE naredba omogućava ponovo očitavanje vrednosti iz **DATA** naredbi.

Izvršavanjem naredbe **RESTORE** sledeća **READ** naredba će očitavati i dodeljivati vrednosti od prve vrednosti prve **DATA** naredbe u programu.

Primer:

```
10 READ A,B,C
20 RESTORE
30 READ D,E,F
40 PRINT A,B,C,D,E,F
50 DATA 12,23,71
```

GOTO n

Naredbom **GOTO** n (ili **GO TO** n) bezuslovno se prelazi na izvršavanje naredbi u programskoj liniji n.

Ukoliko navedena linija ne postoji, prijaviće se izveštaj o grešci **?UNDEF'D STATEMENT.**

Primer:

```
10 X=1
20 PRINT X:X=X+1
30 GOTO 20
```

Izvršavanjem ovog programa na ekranu se ispisuje kolona brojeva počevši od broja 1.

GOTO naredba može biti upotrebljena direktno za ulazak u program na željenom mestu. To može biti od koristi u toku razvijanja programa.

Upotrebom naredbi **ON ... GOTO** može se ostvariti grananje na različite programske linije, u zavisnosti od rezultata navedenog izraza (videti naredbu **ON**).

GOSUB n, RETURN

Naredbom **GOSUB** odlazi se u potprogram, a naredbom **RETURN** vraća se iz potprograma.

U programima često se javlja potreba da se grupa naredbi izvrši više puta. Da se svaki put kada je to potrebno taj deo ne bi ponavljao u tekstu programa, on se izdvaja i naziva potprogramom. Do potprograma se stiže naredbom **GOSUB** n gde je broj n broj prve linije u potprogramu.

Poslednja naredba u potprogramu uvek je **RETURN** (treba je razlikovati od tastera RETURN). Njenim izvršavanjem vraća se iz potprograma u glavni program i prelazi na izvršavanje prve naredbe iza naredbe **GOSUB** kojom se otišlo u potprogram. Potprogram može sadržati više od jedne **RETURN** naredbe, ako je potrebno vraćanje sa različitih mesta iz potprograma.

Potprogram može biti pozivan proizvoljan broj puta u programu. Takođe može biti pozivan iz nekog drugog potprograma. Jedino ograničenje je raspoloživa memorija. Svaki put kada se izvrši naredba **GOSUB** računar, u delu memorije koji se naziva stek, beleži broj programske linije i položaj **GOSUB** naredbe u njoj. Stek je veličine 256 bajta i time ograničava broj potprograma koji su istovremeno pozvani.

Potprogram može biti smešten bilo gde u programu. Da bi se sprečilo slučajno ulaženje u potprogram preporučuje se upotreba **STOP, END** ili **GOTO** naredbe ispred potprograma da preusmeri tok programa.

Upotrebom **ON ... GOSUB** naredbi može se ostvariti grananje na različite potprograme u zavisnosti od dobijenog rezultata.

Dat je primer upotrebe potprograma.

Primer:
```
10 GOSUB 100
20 PRINT "POVRATAK IZ POTPROGRAMA"
30 END
100 PRINT "IZVRSAVA"
110 PRINT "SE"
120 PRINT "POTPROGRAM"
130 FOR N=1 TO 800:NEXT N
140 RETURN
```

GOSUB naredbom u liniji 10 poziva se potprogram koji počinje linijom 100. Time je izvršeno grananje u programu na liniju 100 odakle se izvršavaju naredbe do **RETURN** naredbe u liniji 140. Tada se izvršavanje programa nastavlja od sledeće naredbe iza **GOSUB**. U primeru to je linija 20. **END** naredba u liniji 30 sprečava izvršavanje potprograma po drugi put.

Prilikom pravljenja velikih programa preporučuje se njihovo razlaganje na manje celine, module. Time se dobija na preglednosti i olakšava programiranje i testiranje. Ako se takvi moduli pozivaju više puta, oni se prave u obliku potprograma.

IF

Ovom naredbom donose se odluke koje određuju dalji tok programa, u zavisnosti od rezultata navedenog uslova.

Oblici u kojima se može primenjivati su:

IF uslov **GOTO** broj linije
IF uslov **THEN** broj linije
IF uslov **THEN** naredba

Ako je uslov ispunjen, prelazi se na programsku liniju čiji je broj naveden iza **GOTO** ili **THEN**, ili se nastavlja sa izvršavanjem naredbi navedenih iza **THEN**, uključujući sve naredbe do kraja linije. Ako uslov nije ispunjen, prelazi se na sledeću programsku liniju.

Uslov se zadaje u obliku izraza. Može sadržati brojeve, stringove, promenljive, relacione operatore (=,<,>,<=,>=,<>) i logičke operatore (**NOT, AND, OR**). Ako je uslov ispunjen, tj. tačan, rezultat izraza biće različit od nule, a ako uslov nije ispunjen, tj. nije tačan, rezultat će biti nula.

Primer:

```
10 GET A$
20 IF A$="X" THEN PRINT A$;
30 GOTO 10
```

U liniji 10 očitava se stanje na tastaturi. Ako je pritisnut taster X (uslov A$="X"), u liniji 20 ispisuje se karakter pritisnutog tastera, tj. slovo X.

IF naredba omogućava formiranje petlje u programu (uz grananje, petlja je osnovna programska struktura).

Primer:

```
10 A=0
20 PRINT A,A*A
30 A=A+1
40 IF A<11 GOTO 20
50 END
```

Program ispisuje vrednosti A i A^2 kada se A menja od 0 do 10. U liniji 30 vrednost A se uvećava za jedan. Linija 40 dovodi do ponavljanja prethodnih naredbi sve dok je A<11.

Primer:

```
10 PRINT"(CLR)"
20 PRINT "ZELITE LI OVAJ PRIMER ?"
30 PRINT "ODGOVORITE SA DA ILI NE"
40 INPUT A$
50 IF A$="DA" THEN PRINT TAB(16) "SUPER":END
60 IF A$="NE" THEN PRINT "BAS STE LENJI":END
70 GOTO 40
```

Ako je odgovor bio DA, treba izbrisati prethodni primer i uneti sledeći.

Primer:

```
10 PRINT "(CLR)"
20 A$="G"
30 B$="GRRRRRRRRRRRRRRR"
40 PPINT TAB(5)B$
50 A$=A$+"R"
60 PRINT TAB(5)A$ :PRINT:PRINT"(C/UP)(C/UP)"
70 FOR N=1 TO 10:NEXT N
80 GET C$:IF C$="" GOTO 50
90 PRINT:PPINT
100 IF A$<B$ THEN PRINT"KRATKO":END
110 IF A$>B$ THEN PRINT"DUGO":END
120 PRINT "BRAVO"
```

Ovo je jedan mali test brzine reagovanja. Posle startovanja programa, pritiskom na bilo koji taster treba zaustaviti ispisivanje, tako da ono što se ispisuje bude iste dužine kao i ono što je već napisano.

U liniji 10 unutar navodnika je unesen karakter za brisanje sadržaja ekrana (SHIFT CLR/HOME), a u liniji 60 kontrolni karakter za pomeranje kursora naviše. Linija 70 je uvedena da uspori ispisivanje, što omogućava da se promenom broja 10 menja težina testa.

ON

Naredbom **ON** ostvaruje se grananje na jednu ili više programskih linija u zavisnosti od vrednosti izraza.

Može se primenjivati u oblicima:

ON n **GOTO** br. linije, br. linije, br. linije,...
ON n **GOSUB** br. linije, br. linije, br. linije,...

gde n mora biti brojni izraz. Vrednost izraza zaokružuje se na manji ceo broj ako nije celobrojna. Vrednost mora biti u opsegu od 0 do 255. U protivnom prijavljuje se izveštaj o grešci **?ILLEGAL QUANTITY.** Ako je vrednost n=0 ili je veća od ukupnog broja linija predviđenih za grananje (ali manje ili jednako 255) program će se nastaviti sledećom naredbom u programu.

Brojevi linija (br. linije) na koje se želi grananje u programu navode se međusobno odvojeni zarezima iza **GOTO,** odnosno **GOSUB.**

Vrednost izraza određuje na koju liniju će se izvršiti grananje. Na primer, ako je vrednost izraza 3, izvršiće se grananje, odnosno prelazak, na liniju čiji je broj treći po redu. Maksimalan broj linija je 255.

Primer:
```
10 INPUT X
20 ON X GOTO 40,50,60
30 GOTO 10
40 PRINT "PRITISNUT JE TASTER 1":END
50 PRINT "PRITISNUT JE TASTER 2":END
60 PRINT "PRITISNUT JE TASTER 3":END
```

ON naredba se koristi kao zamena za višestruke IF naredbe.

Primer: **140 ON A-1 GOTO 190, 230, 450, 700**

Program se nastavlja od linije 190 ako je A-1 jednako 1, od linije 230 ako je A-1 jednako 2, od linije 450 ako je A-1 jednako 3, i od linije 700 ako je A-1 jednako 4. Ako je A-1 nula ili veće od 4 nema grananja i program se nastavlja sledećom linijom iza linije 140.

FOR, TO, STEP, NEXT

Ove naredbe koriste se za formiranje petlje, kada je potrebno jednu ili više naredbi izvršiti željeni broj puta.

Petlja, osnovna programska struktura za višestruko izvršavanje naredbi, može se ostvariti **FOR...NEXT** naredbama. Prva naredba u petlji je u opštem obliku:

FOR n=a **TO** b **STEP** c

n je promenljiva koja se koristi kao brojač petlje i naziva se indeks petlje.
a je početna vrednost indeksa.

b je krajnja vrednost indeksa.
c je korak promene indeksa.

a, b, c su brojne vrednosti koje mogu biti zadate i u obliku promenljivih ili izraza. Ne moraju biti celobrojne, a mogu biti i negativne.

Ako se **STEP** i vrednost c izostave, korak promene će biti 1.

Poslednja naredba u petlji je oblika **NEXT** n, pri čemu se n može izostaviti, čime će se dobiti brže izvršavanje petlje.

Primer:
```
10 FOR N=0 TO 10 STEP .5
20 PRINT N,N*N
30 NEXT N
```

Izvršavanjem ove petlje indeks N se menja od 0 do 10 sa korakom 0.5. U svakom krugu izvršavanja ispisuju se vrednosti N i N^2.

Primer: Ako se linija 10 izmeni u:

10 FOR N=0 TO 10

korak promene indeksa petlje N će biti 1.

Ako je početna vrednost indeksa manja od krajnje vrednosti, a korak promene negativan, ili ako je početna vrednost indeksa veća od krajnje vrednosti, a korak promene pozitivan, petlja će se izvršiti samo jednom, a zatim će se preći na izvršavanje sledeće naredbe iza **NEXT**.

Petlja se može napustiti i pre nego što indeks dostigne krajnju vrednost. Takođe je dozvoljeno i vraćanje u petlju. Pri tome treba obratiti pažnju na vrednost indeksa. On se može menjati i u petlji i van petlje, ali pri tome treba biti oprezan. Ne preporučuje se!

Petlja se koristi i za formiranje kašnjenja.

Primer:
```
10 FOR I=1 TO 200
20 PRINT I
30 GET A$:IF A$="S" GOTO 100
40 NEXT I:PRINT "ZAVRSIO SAM":END
100 PRINT " STAO SAM"
110 FOR J=1 TO 400:NEXT J
120 PRINT " SAMO NA KRATKO":GOTO 40
```

U ovom programu postoje dve petlje. U prvoj se njen indeks I menja od 1 do 200. Pri tome se ispisuje vrednost indeksa. Pritiskom na taster S petlja se napušta i ulazi se u drugu sa indeksom J, pomoću koje se ostvaruje pauza. Po njenom završetku vraća se u prvu petlju.

Uobičajena je upotreba petlje u kojoj se nalazi druga petlja. To je dvostruka petlja. Takođe se koriste i višestruke petlje. Pri njihovoj upotrebi petlje se ne smeju preklapati, već jedna mora biti u drugoj, tj. početak i kraj jedne petlje moraju biti između početka i kraja druge.

Primer:
```
10 FOR A=0 TO 9
20 PRINT
30 FOR B=1 TO 5
40 PRINT TAB(4*B)A*5+B;
50 NEXT B:NEXT A
```

Petlja sa indeksom B nalazi se u petlji sa indeksom A. Program ispisuje na ekranu matricu brojeva.

Liniju 50 moguće je napisati i u sledeća dva oblika:

```
50 NEXT B,A
50 NEXT:NEXT
```

Pri tome drugi oblik obezbeđuje brže izvršavanje petlji.

DIM

Naredba **DIM** rezerviše prostor u memoriji za smeštanje vrednosti višedimenzionalnih brojnih i string promenljivih (vektora i matrica).

Iza reči **DIM** navodi se ime, a zatim se u zagradi daju dimenzije te višedimenzionalne promenljive.

Primer: `10 DIM E(3)`

Ovom naredbom se definiše vektor (jednodimenzionalna promenljiva, tj. niz brojeva) sa imenom E, koji se sastoji od sledećih elemenata: E(0), E(1), E(2) i E(3). Svaki od tih elemenata je posebna promenljiva.

Primer: `10 DIM F$(4)`

Ovom naredbom se definiše vektor (niz stringova) koji se sastoji od pet elemenata: F$(0), F$(1), F$(2), F$(3) i F$(4).

Navođenjem znaka procenta (%) iza imena promenljive definisaće se višedimenzionalna celobrojna promenljiva.

Pri definisanju matrica (promenljive sa dve i više dimenzija) dimenzije treba razdvojiti zarezom.

Primer: `10 DIM G(3,4,5)`

Na ovaj način je definisana trodimenzionalna matrica koja sadrži 120 brojnih promenljivih. Označavaju se i predstavljaju na sledeći način:

G(0,0,0), G(0,0,1),..., G(0,0,5)
G(0,1,0), G(0,1,1),..., G(0,1,5)

G(1,0,0), G(1,0,1),..., G(1,0,5)
G(1,1,0), G(1,1,1),..., G(1,1,5)

G(2,3,0), G(3,4,1),..., G(3,4,5)

Istim načinom, ali uz navođenje odgovarajućih imena promenljivih, definišu se celobrojne i string matrice.

Broj dimenzija može biti maksimalno 255, a broj elemenata, tj. promenljivih po dimenziji može biti 32767. Praktično ograničenje koje se može javiti je veličina raspoložive RAM memorije. Dimenzije višedimenzionalne promenljive mogu biti zadate i u obliku promenljivih i izraza, a ako nisu cele vrednosti, zaokružuju se na manju celu vrednost.

Rezervisanje memorije izvršenjem naredbe DIM je sledeće:

5 bajta za ime višedimenzionalne promenljive
2 bajta po dimenziji

2 bajta po elementu za celobrojne promenljive
5 bajta po elementu za realne brojne promenljive
3 bajta po elementu za string promenljive
1 bajt po karakteru string elementa.

Izvršenjem naredbe elementi višedimenzionalne promenljive dobijaju vrednost 0 ako se radi o brojnim promenljivama, odnosno postaju prazan string, ako se radi o string promenljivama.

Naredba DIM se može zadati samo programski, i mora biti zadata samo jedanput po jednoj višedimenzionalnoj promenljivoj. U protivnom prijaviće se izveštaj o grešci **REDIM'D ARRAY.**

Dodeljivanje vrednosti elementima vektora i matrica obavlja se uobičajenim naredbama za dodelu vrednosti promenljivama.

Primer: Dodeljivanje vrednosti 20 četvrtom elementu (indeks 3) vektora E.

```
10 E(3)=20
```

Ako se u programu upotrebi višedimenzionalna promenljiva za koju nije prethodno izvršena naredba DIM, to će se automatski izvršiti za po 11 elemenata po dimenziji. Izvršivši gornji primer i zadajući naredbu **PRINT E(3)** dobiće se odgovor 20. Za ostale elemente sa indeksima od 0 do 10 dobiće se odgovor 0, a za indekse 11 i više izveštaj o grešci **?BAD SUBSCRIPT.**

U slučaju matrica potrebno je iza imena navesti indekse elementa tj. promenljive kojoj se dodeljuje vrednost. Indeksi moraju biti unutar zagrada i međusobno razdvojeni zarezima.

Primer:

```
10 H$(2,3,5)="CBM"
```

U ovom primeru se elementu trodimenzionalne string matrice H$, sa indeksima 2, 3 i 5 dodeljuje vrednost CBM.

Dat je primer dodeljivanja vrednosti dvodimenzionalnoj string matrici sa tri puta tri elementa očitanih iz datoteke. Ostvareno je i njihovo ispisivanje u cilju preglednosti.

Primer:

```
10 DIM A$(2,2)
20 FOR N=0 TO 2:FOR M=0 TO 2
30 READ A$(N,M)
40 PRINT TAB(4*M)A$(N,M);
50 NEXT:PRINT:NEXT
60 DATA "A","B","C","D","E","F","G","H","IJK123+-"
```

4.2.2 Aritmetičke operacije

Komodorov bejzik omogućava sledeće aritmetičke operacije:

+ sabiranje
− oduzimanje
* množenje
/ deljenje

Ove operacije se obavljaju između dve brojne vrednosti. Operacije + i − mogu da stoje ispred samo jedne vrednosti. Osim u brojnim obliku, vrednosti mogu biti zadate i u obliku promenljivih ili izraza.

Primer: naredba ekran
PRINT 2+3 5

Primer:
```
10 X=2:Y=5
20 PRINT X+Y,X-Y,X*Y,X/Y
```

Množenje i deljenje su višeg prioriteta pa se u izrazima izvršavaju pre sabiranja i oduzimanja, osim ako nisu upotrebljene zagrade.

4.2.3 **Funkcije**

Komodorov bejzik omogućava izračunavanje više funkcija. Pri tome, argumenti funkcija su brojne vrednosti koje mogu biti zadate i u obliku promenljivih i izraza.

SQR(x)

SQR (x) Kvadratni koren ($\sqrt{x}$); x mora biti veće od nule ili jednako nuli ($x \geq 0$). Za x manje od nule ($x<0$) dolazi do greške sa izveštajem **?ILLEGAL QUANTITY.**

Primer:
```
10 PRINT SQR(2)
```

LOG (x)

Prirodni logaritam, tj. logaritam za osnovu e (ln x); x mora biti veće od nule ($x>0$), u protivnom javlja se greška sa izveštajem **?ILLEGAL QUANTITY.** Pomoću ove funkcije može se odrediti vrednost logaritma za bilo koju osnovu, i to pomoću izraza:

$$\log_a x = \log x / \log a$$

Za dekadni logaritam: $\log_{10} x = \log x / \log 10$.

Primer: Izračunavanje prirodnog i dekadnog logaritma broja 9.
```
10 PRINT LOG(9),LOG(9)/LOG(10)
```

EXP (x)

Eksponencijalna funkcija (e^x); e=2.71828183. Za vrednosti argumenta x većeg od 88.0296919 javlja se greška sa izveštajem **?OVERFLOW.**

Primer:
```
10 PRINT EXP(1),EXP(4)
```

x↑y

Stepenovanje (x^y);

Primer:
```
10 PRINT 2^3
```

INT (x)

Zaokruživanje na manju celobrojnu vrednost.

Primer:
```
10 PRINT INT(3.7),INT(-4.1)
```

Obratiti pažnju da je **INT (−4.1) = −5.**

ABS (x)

Apsolutna vrednost (|x|). Negativne vrednosti postaju pozitivne, a pozitivne ostaju pozitivne. Dobija se broj bez predznaka.

Primer: `10 PRINT ABS(-9.3),ABS(0),ABS(4.9)`

SGN(x)

Signum funkcija (sgn x). sgn x = 1; x >0
sgn x = 0; x = 0
sgn x = −1; x <0

Primer: `10 PRINT SGN(-9.3),SGN(0),SGN(4.9)`

π

Funkcija bez argumenta koja daje odnos obima i prečnika kruga (3, 14159265).

Primer: `10 PRINT π,π^2`

4.2.4 Trigonometrijske funkcije

Trigonometrijske funkcije kojima raspolaže Komodor su:

SIN (x)

Funkcija nalazi sinus ugla x datog u radijanima (1 radijan $=180/\pi$ stepeni, odnosno $360° = 2*\pi$ radijana).

Primer: `10 PRINT SIN(20),SIN(25*π/180)`

Izračunava se sinus ugla od 20 radijana i sinus ugla od 25 stepeni.

COS (x)

Funkcija nalazi kosinus ugla x datog u radijanima.

Primer: `10 PRINT COS(20),COS(25*π/180)`

Izračunava se kosinus ugla od 20 radijana i kosinus ugla od 25 stepeni.

TAN (x)

Funkcija nalazi tangens ugla datog u radijanima. Za vrednost ugla bliske ili jednake $\pi/2$ ili $-\pi/2$ javiće se izveštaj o grešci **?DIVISION BY ZERO.**

Primer: `10 PRINT TAN(0),TAN(π/3),TAN(π/2)`

ATN (x)

Funkcija nalazi arkustangens brojne vrednosti x. Rezultat je u radijanima i uvek je između $-\pi/2$ i $\pi/2$.

Primer:
```
10 PRINTATN(-123456),ATN(0),ATN(123456)
```

4.2.5 Operacije poređenja

Komodor omogućava sledeće operacije poređenja:

= jednako
< manje
> veće
< = manje ili jednako
> = veće ili jednako
<> različito

Relacioni operatori (=,<,>,<=,>=,<>) mogu se koristiti u izrazima sa brojnim i string vrednostima. Time se omogućuje kako poređenje brojnih vrednosti, tako i string vrednosti. Najčešće se koriste u okviru naredbe za grananje u programu na osnovu zadatog uslova (naredba IF).

Operacije poređenja rezultuju brojnom vrednošću. Rezultat operacije je −1 ako je ishod poređenja tačan, a ako nije tačan rezultat je 0.

Primeri:

naredba	rezultat
PRINT 5>3	**−1**
PRINT 5<3	**0**

Primer: Ispisuju se rezultati svih poređenja dve brojne vrednosti, koje se unose pomoću **INPUT** naredbe.

```
10 INPUT "X";X:INPUT"Y";Y
20 PRINTX=Y,X<Y,X>Y,X<=Y,X>=Y,X<>Y
```

Poređenje stringova se obavlja tako što se porede kodovi karaktera. Pri tome je karakter sa većom vrednošću koda „veći".

Primeri:

naredba	rezultati
PRINT "A">"B"	**0**
PRINT "A"<"B"	**−1**

U slučaju stringova sa više od jednog karaktera obavlja se poređenje prvih karaktera. Ako su oni jedaki porede se drugi karakteri i tako dalje.

Primeri:

naredba	rezultat
PRINT "AAB">"AB"	**0**
PRINT "AH"<"IH"	**−1**

U operacijama poređenja moguće je porediti brojne vrednosti samo sa brojnim vrednostima. Isto važi i za stringove. U protivnom javlja se greška sa izveštajem **?TYPE MISMATCH.**

Pogodna upotreba operacija poređenja omogućava pravljenje kraćih i bržih programa.

Primer:

```
10 FOR N=0 TO 20
20 PRINT TAB((N>10)*(10-N))"+"
30 NEXT
```

4.2.6 Logičke operacije

Logičke operacije koriste se uz operacije poređenja utičući na značenje poredbenih operatora i rezultat poređenja. Takođe se koriste za obavljanje Bulove algebre nad navedenim argumentima. Pri tome argumenti moraju imati vrednosti između −32768 i +32767. Ako nisu celobrojne, zaokružuju se na manju vrednost.

x AND y

Operacija logičkog množenja (logičko I). Koristi se za proveru tačnosti oba argumenta.

Primer: **IF A=2 AND B=3 GOTO 100**

Grananje u programu na liniju 100 izvršiće se ako je A=2 i (engl. and) B=3.

Upotreba logičkog operatora **AND** kao Bulovog operatora, kako se često naziva, vidi se iz tabele istinitosti.

0 **AND** 0=0
0 **AND** 1=0
1 **AND** 0=0
1 **AND** 1=1

Izvršavanjem operacije **AND** izvršiće se logičko množenje, prema gore navedenim pravilima, nad bitima binarno predstavljanih argumenata.

Primer:	naredba	rezultat
	PRINT 9 AND 12	**8**

Predstavljanjem argumenata 9 i 12 u binarnom obliku i izvršenjem Bulovog množenja dobija se rezultat 8.

```
      9          1001
AND  12     AND  1100
-------     ---------
      8          1000
```

x OR y

Operacija logičkog sabiranja (logičko ILI). Koristi se za proveru tačnosti barem jednog argumenta.

Primer: **IF A=2 OR B=3 GOTO 100**

Grananje u programu na liniju 100 izvršiće se ako je A=2 ili (engl. or) B=3.

Upotreba logičkog operatora **OR** kao Bulovog operatora se obavlja prema sledećoj tabeli.

0 **OR** 0=0
0 **OR** 1=1
1 **OR** 0=1
1 **OR** 1=1

Izvršavanjem operacije **OR** izvršiće se logičko sabiranje prema gore navedenim pravilima, nad bitima binarno predstavljenih argumenata.

Primer: naredba **PRINT 9 OR 12** rezultat **13**

Predstavljanjem argumenata 9 i 12 u binarnom obliku i izvršenjem Bulovog sabiranja dobija ie rezultat 13.

```
     9              1001
OR  12         OR   1100
    13              1101
```

NOT x

Operacija logičke negacije (logičko NE). Koristi se za promenu tačnosti argumenta. Izvršava se samo nad jednim argumentom.

Primer: **IF NOT A<>B GOTO 100**

Grananje u programu na liniju 100 izvršiće se ako A nije (engl. not) različito od B. Odnosno ako je A jednako B.

Za upotrebu logičkog operatora **NOT** kao Bulovog operatora, važi sledeće:

NOT 0=1
NOT 1=0

Izvršavanjem operacije **NOT** izvršiće se logička negacija, prema gore navedenim pravilima, nad bitima binarno predstavljenih argumenata.

Primer: naredba **PRINT NOT 9** rezultat −10

Rezultat −10 predstavlja brojnu vrednost u komplementu dvojke (videti poglavlje 7), što znači da važi izraz:

NOT x = −(x+1)

Predstavljanje argumenta 9 u binarnom obliku i izvršenjem Bulove negacije dobija se rezultat −10 u komplementu dvojke.

```
NOT 9          NOT 0000000000001001
  −10              1111111111110110
```

4.2.7 Prioriteti funkcija i operacija

Funkcije i operacije obavljaju se prema utvrđenom redosledu. Redosled koji određuje koja će se funkcija ili operacija obaviti pre druge naziva se prioritet. Sledeća tabela prikazuje prioritete od najvišeg ka najnižem.

Najviši prioritet 1 sve funkcije
2 stepenovanje
3 predznak minus
4 množenje i deljenje
5 sabiranje i oduzimanje

6 relacije (=,>,<,>=,<=,<>)
7 **NOT**
8 **AND**
Najniži prioritet 9 **OR**

Normalni redosled izvršavanja može biti promenjen upotrebom zagrada (). Sve ono što je navedeno unutar zagrade biće izvršeno pre izvršavanja onoga što je izvan zagrada. Unutar zagrada se mogu nalaziti druge zagrade. Pri tome je redosled izvršavanja od unutrašnjih ka spoljnim zagradama. Neophodno je da broj levih i broj desnih zagrada bude isti. U protivnom javlja se greška sa izveštajem **?SYNTAX ERROR.**

DEF FN

Ovom naredbom se definišu i daju imena funkcijama.

U programu se neki izraz može ponavljati na više mesta. Da bi se uštedela memorija, naročito kod dugih izraza, definiše se funkcija kojoj se dodeljuje vrednost izraza.

Opšti oblik naredbe za definisanje funkcije je:

DEF FN a(b)=izraz

a je ime funkcije. Mora se sastojati od jednog ili dva karaktera. Prvi mora biti slovo, a drugi slovo ili broj.

b je argument funkcije. To je ime brojne promenljive, koja će u definiciji funkcije biti zamenjena brojnom vrednošću, kada se bude izračunavala funkcija. Mora biti formalno naveden u slučaju da ne postoji u izrazu. Izraz se navodi sa desne strane znaka jednakosti. On određuje vrednost funkcije i može biti samo brojni izraz sa jednim argumentom.

Argument b se upotrebljava samo u definiciji funkcije, ne utičući na programsku promenljivu sa istim imenom. Argument se ne mora nalaziti u izrazu. Ako se nalazi, pri izračunavanju funkcije (naredbom **FN**) uzima se njegova vrednost. U protivnom uzima se tekuća vrednost promenljive.

Naredba **DEF FN** mora biti izvršena, da bi definisala funkciju, pre izračunavanja funkcije. U protivnom prijaviće se izveštaj o grešci **? UNDEF'D FUNCTION.** Funkcija može biti više puta definisana u programu. Tada se pri izračunavanju funkcije koristi poslednja definicija.

DEF FN ne može se koristiti u direktnom načinu rada.

Primer:
```
10 PI-3.1415
20 OEF FN PO(D)-O^2*PI/4
30 INPUT"PRECNIK";PR
40 PRINT "POVRSINA JE" FN PO(PR)
```

U liniji 20 definiše se funkcija **FN PO** koja određuje površinu kruga prečnika D. Funkcija se izračunava u liniji 40.

FN

Izračunavanje vrednosti funkcije definisane naredbom **DEF FN** ostvaruje se naredbom **FN.**

FN a(c)

a je ime prethodne definisane funkcije.

c je brojna vrednost, koja može biti zadata i u obliku promenljive ili izraza.

FN se može koristiti i u direktnom načinu rada, ako je izvršeno prethodno definisanje funkcije.

Primer: Videti primer za **DEF FN** naredbu.

RND (n)

Pomoću **RND** dobija se slučajni broj između 0 i 1.

U zagradi, iza reči **RND**, navodi se brojna vrednost n, koja može biti data i u obliku promenljive ili izraza. Ona utiče na dobijanje slučajnog broja na sledeći način.

Ako je argument n pozitivan, ne dobijaju se pravi slučajni brojevi već tzv. pseudoslučajni brojevi. Oni čine niz od 65535 različitih brojeva koji počinju da se ponavljaju posle 65535 brojeva. Početna vrednost je tzv. osnova (engl. seed) i postavlja se uključenjem računara.

Ako je argument n jednak nuli, tada se slučajni broj dobija direktno iz ugrađenog časovnika.

Ako je argument n negativan, dobija se slučajni broj uz prethodno postavljanje osnove, za navedeni argument.

Primena slučajnih brojeva je široka, od igara do raznih stručnih disciplina.

Primer:
```
10 FOR N=1 TO 10
20 PRINT RND(0)
30 NEXT N
```

U ovom primeru ispisuje se deset slučajnih brojeva u opsegu od 0 do 1.

Primer: Na sledeći način može se dobiti 7 slučajnih brojeva u opsegu od 1 do 40, što odgovara izvlačenju brojeva u igri LOTO.

```
10 FOR N=1 TO 7
20 PRINT 1+INT(40*RND(1))
30 NEXT N
```

U datom primeru može doći do ponavljanja brojeva, što ne odgovara igri LOTO. Predlaže se, u cilju vežbe, proširenje datog primera tako da realno odgovara igri.

4.2.8 Rad sa stringovima

Sabiranje stringova (+)

Sabiranje stringova A i B dobija se string koji se sastoji od stringa A i nadovezanog stringa B.

Sabiranje stringova je jedina operacija koja se može izvršiti sa dva stringa. Stringovi se sabiraju tako što se između njih postavi operator +.

Primer:	naredba	rezultat
	PRINT „AB"+„C2E"	**ABC2E**

Umesto stringova mogu se koristiti string promenljive i izrazi.

Primer:
```
10 A$="BEO":B$="GRAD"
20 PRINT A$+B$,B$+A$
```

Važno je uočiti da u slučaju zamene mesta stringova, u operaciji sabiranja, rezultat nije isti.

Podstringovi

Podstring je deo stringa. Sastoji se od određenog broja susednih karaktera nekog stringa. Sledeće tri naredbe služe za dobijanje podstringova.

LEFT$ (x$,n)

Ovom naredbom dobijaju se prvih n karaktera stringa x $.

Primer:
```
10 A$="MIKRO KNJIGA"
20 PRINT LEFT$(A$,5)
```

U naredbi string može biti zadat neposredno, u obliku promenljive ili izraza. Isto važi i za broj n. On mora biti u opsegu od 0 do 255. Ako nije ceo broj zaokružuje se na manju celobrojnu vrednost. Ako je veći ili jednak broju karaktera u stringu rezultat je ceo string, a ako je 0, rezultat je prazan string.

RIGHT$ (x$, n)

Ovom naredbom dobijaju se poslednjih n karaktera stringa x $.

Primer:
```
10 A$="MIKRO KNJIGA"
20 PRINT RIGHT$(A$,6)
```

String može biti zadat neposredno, u obliku promenljive ili izraza. Isto važi i za broj n. On mora biti u opsegu od 0 do 255. Ako nije ceo broj zaokružuje se na manju celobrojnu vrednost. Ako je veći ili jednak od broja karaktera u stringu rezultat je ceo string, a ako je 0 rezultat, je prazan string.

MID$ (x$,n,m)

Ovom naredbom dobija se željeni deo stringa x $. Broj n određuje položaj prvog karaktera podstringa u stringu, a m željenu dužinu.

Primer:
```
10 A$="MIKRO KNJIGA"
20 PRINT MID$(A$,3,6)
```

Rezultat je string KRO KN. On je podstring stringa MIKRO KNJIGA.

String može biti zadat neposredno, u obliku promenljive ili izraza. Isto važi i za brojeve n i m. Oni moraju biti u opsegu od 0 do 255. Ako nisu celi zaokružuju se na manju celobrojnu vrednost. Ako je n veće od dužine (broja karaktera) stringa ili je m nula, rezultat je prazan string. Ako se m izostavi uzeće se dužina stringa do kraja. Isto će se desiti ako je m veće od broja preostalih karaktera u stringu.

Primer: Program prikazuje sve podstringove stringa MIKRO KNJIGA.

```
10 A$="MIKRO KNJIGA"
20 FOR N=1 TO 12:FOR M=1 TO 13-N
30 PRINT MID$(A$,N,M)
40 NEXT M,N
```

CHR$ (n)

Ovom naredbom se prevodi Komodorov kôd (PETASCII) u odgovarajući karakter.

Svakom karakteru (slovo, broj, znak,...) pridodat je jedan broj koji se naziva kôd karaktera. Tabela sa kodovima i karakterima je data u dodatku.

Primer: Kôd slova A je 65, pa je rezultat naredbe **PRINT CHR$(65)** slovo **A.**

Kôd može biti zadat i u obliku promenljive ili izraza. Ako nije celobrojna vrednost zaokružuje se na manju celobrojnu vrednost. Ako je manji od 0 ili veći od 255 javiće se greška sa izveštajem **?ILLEGAL QUANTITY.**

Primer: Pomoću sledećeg programa prikazuju se karakteri čiji su kodovi između 32 i 127

```
10 FOR N=32 TO 127
20 PRINT CHR$(N);
30 NEXT N
```

Kontrolni kodovi takođe mogu biti upotrebljeni naredbom **CHR$.**

Primer:
```
10 PRINT CHR$(147)
```

Kôd 147 je kontrolni kôd SHIFT CLR/HOME čime se ostvaruje brisanje sadržaja ekrana.

ASC (n$)

Ovom naredbom nalazi se brojna vrednost—kôd prvog karaktera stringa n$.

String može biti zadat u obliku promenljive ili izraza. Rezultat je celobrojna vrednost između 0 i 255.

Primer:

naredba	rezultat
PRINT ASC („ABCD")	**65**

Ako je string prazan, tj. ne sastoji se oni od jednog karaktera pojaviće se greška sa izveštajem **?ILLEGAL OUANTITY**. U slučaju da je potrebno primenjivati **ASC** naredbu nad stringom koji može biti i prazan potrebno mu je dodati **CHR$(0)**, što je karakter praznog stringa.

LEN (x$)

Pomoću **LEN** dobija se ukupan broj karaktera u stringu x $.

Primer:

naredba	rezultat
PRINT LEN(„ABCD")	**4**

String može biti i u obliku promenljive ili izraza.

Primer:
```
10 INPUT "UPISITE VASE IME I PREZIME";A$
20 FOR N=1 TO LEN(A$)
30 PRINT LEFT$(A$,N)
40 NEXT N
```

STR$ (n)

Naredba **STR $** prevodi brojnu vrednost u string. Karakteri stringa su cifre navedenog boja.

Primer:

naredba	rezultat
PRINT STR$ (299792)	299792
ovo je broj	ovo je string

Primer:

```
10 A$=STR$(12)
20 PRINT A$
30 B$=STR$(104-2.31)
40 PRINT B$
50 PRINT A$+B$
```

VAL (n$)

Prevodi string u brojnu vrednost.

Ako prvi karakter stringa, koji nije prazno polje, nije cifra ili predznak plus (+) ili minus (−) rezultat je nula.

naredba	rezultat
PRINT VAL („2+3.1")	**5.1**
PRINT VAL („12E2G3")	**1200**

String može biti dat i u obliku promenljive i izraza.

Primer:

```
10 A$="2.99792E5"
20 PRINT VAL (A$)
```

.2.9 Ostale naredbe

TIME

Izvršavanjem ove naredbe očitava se vreme na ugrađenom časovniku u šezdesetim delovima sekunde.

Uključenjem računara vrednost časovnika postavlja se na nulu. Svakih šezdeset delova sekunde (1/60 s) vrednost časovnika povećava se za jedan. Za vreme rada računara sa kasetofonom časovnik je isključen.

TIME se može pisati skraćeno **TI**.

Primer:

```
10 PRINT "OD UKLJUCENJA PROSLO JE"TI/3600"MINUTA
```

TIME$

TIME$ služi za očitavanje i postavljanje realnog vremena.

Unutrašnji časovnik, koji odbrojava vreme sa korakom od šezdesetog dela sekunde (1/60 s), može se očitati pomoću **TIME$**. Pri tome će se dobiti string od šest karaktera, gde su prva dva sati, druga dva minuti i poslednja dva sekunde.

Početna vrednost časovnika takođe se može postaviti pomoću **TIME$**.

U slučaju rada računara sa kasetofonom, časovnik je isključen što će dati pogrešno vreme.

TIME$** se može pisati skraćeno **TI$.

Primer:
```
10 TI$="000000"
20 GET A$:IF A$=""GOTO 20
30 PRINT LEFT$(TI$,2);":";MID$(TI$,3,2);":";RIGHT$(TI$,2)
```

Dati program obavlja funkciju elektronske štoperice. Startovanjem programa, pritiskom na taster RETURN, otpočinje merenje vremena od nula časova, nula minuta i nula sekundi, što je određeno linijom 10. Sve dok se ne pritisne bilo koji taster, što označava kraj merenja, izvršavaće se linija 20. U liniji 30, koristeći naredbe za rad sa stringovima, ispisuje se dvotačka između sati, minuta i sekundi.

FRE (n)

Pomoću **FRE** dobija se broj bajtova u RAM-u, slobodnih za bejzik program i njegove promenljive.

Iza reči **FRE** treba u zagradi navesti bilo koji argument.

Primer: **PRINT FRE (0)**

Ako je dobijeni rezultat negativan potrebno je dodati 65535 da bi se dobila stvarna vrednost. Sledeći primer uvek prikazuje pravu vrednost.

Primer:
```
10 PRINT FRE(0)-65535*(FRE(0)<0)
```

PEEK (n)

Pomoću **PEEK** se očitava sadržaj memorijske lokacije n.

Zadata adresa memorijske lokacije (n) mora biti u opsegu od 0 do 65535. Ako nije zadata celobrojna vrednost zaokružuje se na manju vrednost. Može biti zadata i u obliku promenljivih ili izraza.

U svakoj memorijskoj lokaciji nalazi se jednobajtna vrednost, što znači da će dobijeni rezultat biti celobrojna vrednost između 0 i 255.

Primer: **10 PRINT PEEK (2051)**

Izvršenjem ove programske linije dobija se broj 10, što je sadržaj memorijske lokacije 2051. U njoj se nalazi broj prve linije u programu (bajt manje težine).

POKE n,m

Naredbom **POKE** upisuje se jednobajtna vrednost u memorijsku lokaciju.

n mora biti u opsegu od 0 do 65535 i ukazuje u koju se memorijsku lokaciju upisuje vrednost.

m je jednobajtna vrednost koja se upisuje u memorijsku lokaciju. Mora biti u opsegu od 0 do 255.

Za adresu memorijske lokacije i vrednost koja se u nju upisuje važi da će biti zaokružene ako nisu celi brojevi i da se mogu dati kao promenljive ili izrazi.

Primer: **10 POKE 2051,25**

Nakon izvršenja ove programske linije treba zadati naredbu **LIST** direktno. Programska linija je dobila novi broj, 25 (videti primer za naredbu **PEEK**).

SYS n

Ovom naredbom otpočinje izvršavanje mašinskog programa.

Ovo je jedna od najsnažnijih naredbi bejzika jer omogućava korišćenje velikih mogućnosti mašinskih programa. Izvršavanje mašinskog programa otpočinje od memorijske lokacije čija je adresa data Iza reči SYS. Adresa n može biti zadata i u obliku promenljive ili izraza.

Adresa može biti bilo gde u RAM-u, tj. u opsegu od 0 do 65535. Ako se želi povratak u bejzik, mašinski program se mora završiti mašinskim kodom 96 (RTS-return from subroutine).

Primer: **SYS PEEK (2↑16 – 4) + 256*PEEK(2↑16 – 3)**

U ovom primeru izvršava se reset rutina (mašinski program). Rutina se nalazi na memorijskoj lokacijI koja je određena sadržajem memorijskih lokacija 65532 ($FFFC) i 65532 ($FFFD).

USR (x)

USR je funkcija definisana od strane korisnika u obliku mašinskog potprograma.

Memorijska lokacija na koju se odlazi, radi izvršavanja mašinskog potrpograma, određena je sadržajem memorijskih lokacija 785 (niži bajt) i 786 (viši bajt). Njihov sadržaj potrebno je postaviti pre upotrebe funkcijske naredbe **USR**, na primer naredbom **POKE**. U protivnom, prljaviće se izveštaj o grešci **?ILLEGAL QUANTITY.**

Argument x , navodi se iza reči **USR,** unutar zagrada. Može biti dat i u obliku promenljive ili izraza. Izvršenjem **USR** naredbe stavlja se u akumulator za rad sa realnim brojevima (engl. floating point accumulator), koji počinje od memorijske lokacije 97. Preuzima se od mašinskog potprograma koji izračunava rezultat i smešta ga u isti akumulator pre povratka u bejzik.

Primer:
```
10 POKE 785,234
20 POKE 786,185
30 INPUT X
40 Y=USR(X)
50 PRINT "LOGARITAM OD ";X;" JE";Y
```

U ovom primeru se izračunava logaritam unetog broja. Rutina za izračunavanje logaritma počinje od adrese koja je određena sadržajem memorijskih lokacija 785 i 786. Za potpunije objašnjenje pogledati poglavlje 8.

WAIT

Ova naredba zaustavlja izvršavanje programa sve dok se u određenoj memorijskoj lokaciji vrednost ne postavi na unapred zadatu.

Opšti oblik ove naredbe je:

WAIT lokacija, n, m

lokacija – je adresa bilo koje memorijske lokacije i nalazi se u opsegu od 0 do 65535. Može biti zadata u obliku promenljive ili izraza. Ako nije celobrojna zaokružuje se na manju celu vrednost.

n, m – su celobrojne vrednosti u opsegu od 0 do 255. Mogu se zadati u obliku promenljivih ili izraza. Ako je potrebno zaokružuju se na cele vrednosti. Vrednost m nije obavezna da se navede.

Izvršavanjem naredbe, proverava se stanje u naznačenoj memorijskoj lokaciji. Memorijske lokacije koje se koriste po pravilu su memorijske lokacije kojima mogu pristupiti i spoljne jedinice (tzv. memorijski mapirani ulaz/izlaz, videti poglavlje 8).

Provera stanja lokacije, izvršenjem naredbe **WAIT**, obavlja se na sledeći način. Očitava se vrednost u lokaciji i izvršava se operacija logičkog množenja **(AND)** sa vrednošću n. Ako je u naredbi navedena vrednost m, izvršiće se operacija logičkog isključivog sabiranja između prethodnog rezultata i vrednosti m. Ako je rezultat nula ponovo će se očitavati i proveravati vrednost. Ako je rezultat različit od nule program će se nastaviti sledećom naredbom.

Operacija logičkog isključivog sabiranja (isključivo ILI) obeležava se sa XOR (engl. exclusive or), obavlja se nad dva argumenta i za nju važi sledeće:

0 XOR 0 =0
0 XOR 1 =1
1 XOR 0 =1
1 XOR 1 =0

Operacija XOR obavlja se nad svim bitima odgovarajućih težina (identično operacijama **AND** i **OR**, videti logičke operacije). U naredbi **WAIT**, argument m i izvršenje logičke operacije XOR omogućavaju promenu uslova za zaustavljanje, odnosno nastavljanje izvršavanja programa.

Primer:
```
100 FOR X-40960 TO X+8191:PRINT X,PEEK(X)
110 WAIT 197,64
120 WAIT 197,64,64
130 NEXT
```

Dati program omogućuje pregled sadržaja bejzik ROM-a (linija 100), ali tako da se za svaki pritisak tastera prikazuje sadržaj po jedne memorijske lokacije. Lokacija sa adresom 197 svojim šestim bitom označava da li je neki taster pritisnut ili ne. Ako je bilo koji taster pritisnut šesti bit ima vrednost 1, a ako su svi tasteri otpušteni šesti bit ima vrednost 0. Naredba **WAIT** u liniji 110 zaustavlja program dok prethodni taster ne bude otpušten, a u liniji 120 sve dok neki taster ne bude pritisnut.

4. 2. 10 Rad sa kasetofonom i diskom

Rad sa kasetofonom

Programi i podaci mogu se, u cilju kasnijeg korišćenja, snimiti na magnetnu traku.

Naredbe Komodora koje se odnose na rad sa kasetofonom su: **LOAD, SAVE, VERIFY, OPEN, CLOSE, PRINT#, INPUT# i GET#.** Prve tri su obrađene u daljem tekstu, a ostale u delu o datotekama.

Pre prikaza naredbi izložiće se postavke bitne za što potpunije razumevanje načina snimanja programa i rada kasetofona.

Magnetna traka koja se koristi za snimanje po pravilu je uobičajena muzička kaseta. Prednost kasete namenjene upravo za snimanje programa (tzv. kompjuterske kasete) je u njenoj kraćoj traci, što omogućava brže nalaženje određenog snimka.

Uređaj za snimanje je kasetofon specijalno napravljen za Komodor. PAŽNJA: Prilikom priključivanja bilo kog spoljnog uređaja na računar, što znači i kasetofona, potrebno je sve isključiti iz struje. U protivnom moguća su ozbiljna oštećenja Komodora ili spoljnog uređaja.

Kasetofon se napaja električnom energijom direktno iz Komodora. Uključenjem računara uključuje se i kasetofon. Komande za rad sa kasetofonom identične su komandama običnog kasetofona (**RECORD** – snimanje, **PLAY** – pokretanje trake, **REWIND** – brzo premotavanje nazad, **F. FWD** – brzo premotavanje napred, **STOP/EJT** – zaustavljanje trake i izbacivanje kasete, **PAUSE** – pauza).

Snimanje programa na traku ostvaruje se snimanjem zvučnog zapisa (tona).

Snimak programa na traci sastoji se iz sledećeg:

1. Zaglavlje (engl. header). Prvi bajt zaglavlja određuje vrstu snimka (2 – podaci, 3 – program). Drugi i treći bajt daju adresu memorijske lokacije od koje su program ili podaci snimljeni, tj. adresu na koju će biti učitani. Drugi bajt je bajt manje težine, a treći veće težine. Četvrti i peti bajt određuju adresu memorijske lokacije zadnjeg bajta programa tj. podataka. Naredni bajtovi su za ime programa. Svakom karakteru imena odgovara jedan bajt. Može ih biti do 187.
2. Prvi snimak celog programa. Snima se kompletan program, bejzik ili mašinski.
3. Drugi snimak celog programa.
4. Signal za označavanje kraja trake, tzv. EOT marker (engl. end of tape marker). Izborom naredbe za snimanje određuje se da li će se iza drugog snimka programa snimiti ovaj signal. Omogućuje zaustavljanje trake po učitavanju programa.

Učitavanjem zaglavlja podaci koji se nalaze u njemu smeštaju se u rezervisani deo memorije koji se naziva kasetni bafer (videti poglavlje 8). Očitavanjem sadržaja memorijskih lokacija tog dela memorije (naredbom **PEEK**) može se saznati početak i kraj učitanog programa, što može biti od interesa naročito za mašinske programe, tj. podatke.

Učitavanjem celokupnog programa, sa prvog snimka, obavlja se njegovo smeštanje u određeni deo memorije. Bejzik program smešta se od memorijske lokacije 2049 (početak područja bejzik programa je od 2048). Mašinski programi i podaci smeštaju se od memorijske lokacije koja je određena naredbama za snimanje i učitavanje.

Drugi snimak programa poredi se sa učitanim programom. Ukoliko se pojavi razlika prijaviće se izveštaj o grešci. Greška može biti u prvom ili/i drugom snimku programa. Postavljanjem sistemskih promenljivih VARTAB, ARYTAB i STREND na vrednosti poslednje memorijske lokacije bejzik programa, očitane iz memorije za rad sa kasetofonom, može se pokušati spasavanje bejzik programa.

Do greške u učitavanju može doći zbog nekog oštećenja trake (mehaničko, magnetno, termičko...) ili zbog neke greške u radu kasetofona. Najčešće je u pitanju zaprljanost glave za snimanje i reprodukciju. Njeno čišćenje najbolje je obaviti pomoću vate i alkohola. Trake za čišćenje se ne preporučuju zbog njihovog abrazivnog dejstva.

Rad sa diskom

Disk jedinica je, pored kasetofona, još jedna periferna jedinica za snimanje programa i podataka. Ona ima dve važne prednosti u odnosu na kasetofon:

1. Veća brzina zapisivanja i očitavanja informacija (oko 10 puta).
2. Veća brzina pristupa informacijama i način pristupa informacijama.

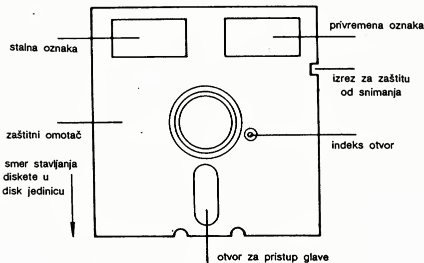
Druga osobina je od prvenstvenog značaja. U radu sa kasetofonom korisnik mora da zna mesto na traci gde se program nalazi (najčešće uz pomoć brojača) da bi ga brzo učitao. U radu sa diskom, sistemski program u ROM memoriji računara i program u samoj disk jedinici, koji se naziva disk operativni sistem ili kraće DOS (engl. Disc Operating System), vode računa i o tome gde se nalaze programi. Time je omogućeno direktno pristupanje

željenom programu. Dobijajući direktni i brži pristup, moguće je na disku organizovati datoteke neostvarive na traci. Naprimer datoteke sa direktnim pristupom.

Komodorova disk jedinica je samostalna jedinica. Ona ima svoje napajanje, svoj mikroprocesor (6502), integralna kola za komunikaciju (6522), 2KB RAM i 16 KB ROM u kome se nalazi DOS. Sa Komodorom se povezuje preko serijske (IEC) veze. Ova veza je slična standardnoj IEEE—488 vezi, ali se podaci šalju serijski, bit po bit. Svaki uređaj povezan preko ove veze mora da poseduje identifikacioni broj (primarnu adresu). Standardni broj za disk jedinicu je 8.

Sa prednje strane disk jedinice nalaze se dve signalne lampice. Zelena pokazuje da li je disk jedinica uključena ili ne. Crvena pokazuje da disk jedinica radi. Ukoliko svetli neprekidno, radi ispravno, a ukoliko se naizmenično pali i gasi, jedinica ne radi ispravno što ukazuje da treba ponoviti traženu operaciju. Sa zadnje strane nalazi se prekidač za uključivanje, priključak za napajanje, osigurač i dva priključka za serijsku vezu. Priključci za vezu su paralelno vezani, tako da se jedan koristi za povezivanje sa računarom, a drugi za dalju vezu sa štampačem, drugom disk jedinicom i sl.

Programi i podaci se pamte na disketi. Disketa je tanka plastična ploča, kružnog oblika, na koju je nanesen feromagnetni materijal. Njena površina je veoma osetljiva i zbog toga se disketa nalazi zaštićena u plastificiranom omotaču.



Sl. 4. 1. Izgled diskete

Na omotaču se nalazi otvor za pristup glave za upisivanje/čitanje disketi. Takođe postoji i indeksni otvor za sinhronizaciju rada disk jedinice. Sa strane se nalazi izrez koji omogućuje snimanje na disketu. Ukoliko se zatvori nalepnicom, snimanje je onemogućeno tj. disketa je zaštićena od slučajnog brisanja. Na disketi se nalaze i dve oznake. Stalna oznaka koju stavlja proizvođač i privremena oznaka koju stavlja korisnik.

Komodorova disk jedinica radi sa disketama od 5 1/4" (pet i četvrt inča) koje su jednostrane (engl. single sided) i sa normalnom tj. jednostrukom gustinom (engl. single density)

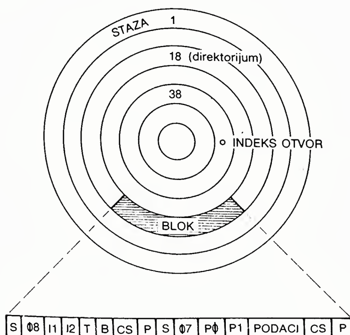
Formatiranje nove diskete

Diskete se mogu koristiti sa različitim disk jedinicama pa ih je, pre njihovog korišćenja potrebno prilagoditi jedinici tj. formatirati. Formatiranje se obavlja izvršavanjem sledećeg programa:

```
10 OPEN 15,8,15
20 PRINT#15,"NEW:IME DISKETE,ID"
30 CLOSE 15
```

Ovaj program je neophodno izvršiti uvek kada se disketa koristi prvi put ili kada celu disketu treba obrisati. Ime diskete bira korisnik i može imati maksimalno 16 karaktera. Identifikacija, id, ima dva proizvoljna karaktera. Pri formatiranju cela površina diskete deli se na staze (engl. track), a zatim se te staze dele na površine, jednake veličine, koje se zovu blokovi. Disketa je podeljena na 35 staza od kojih svaka ima od 17 do 21 blokova.

broj staze	blokova po stazi
1 do 17	21
18 do 24	19
25 do 30	18
31 do 35	17



Sl. 4. 2. Raspored staza i blokova na formatirnoj disketi

Direktorijum (katalog)

Napomenuto je da DOS vodi računa o tome gde se na disketi smeštaju programi i podaci, na koju stazu, u koje blokove i sve te informacije organizuje i čuva u dva posebno iz-

dvojena bloka diskete. Ti blokovi se nazivaju mapa slobodnih blokova (engl. Block Availability Map — BAM) i katalog ili direktorijum (engl. directory).

BAM je lista u kojoj se čuvaju informacije o iskorišćenju svih 683 blokova na disketi i smešten je u bloku 0 na stazi 18. Svaki put kada se neki program upiše na disketu pomoću naredbe SAVE, ili se zatvori neka datoteka pomoću naredbe CLOSE, lista BAM se ažurira listom blokova koji su iskorišćeni za smeštanje tog programa ili datoteke.

Direktorijum je lista svih programa ili datoteka koje su smeštene na jednoj disketi. On je smešten u bloku 1 staze 18. Maksimalni broj elemenata ove liste je 144 tj. na disketu se može snimiti maksimalno 144 programa. Svaki put kada se neki program ili datoteka smesti na disketu, direktorijum se ažurira.

Za prikazivanje sadržaja direktorijuma treba zadati sledeću naredbu:

LOAD "$", 8

Time je direktorijum prebačen u memoriju Komodora i može se videti izvršenjem naredbe LIST.

```
0 „PRIMERI              „01   2A
5 „PRIMER 1"                  PRG
4 „PRIMER 2"                  PRG
1 „PODACI"                    SEQ
654 BLOCKS FREE
```

Prvi red, koji je odštampan inverzno, sadrži ime diskete koje je dodeljeno prilikom formatiranja. U sledećim redovima dat je spisak programa i datoteka. Uz svaki program na početku linije stoji broj blokova diskete upotrebljenih za smeštanje programa ili podataka. Iza svakog imena se nalazi reč koja označava da li se radi o programu (PRG) ili datoteci (SEQ, REL, USR). Poslednji red sadrži broj slobodnih blokova.

Komande diska

Komande diska omogućuju komunikaciju između Komodora i disk jedinice. Komande se šalju preko tzv. komandnog kanala, za koji je rezervisan broj 15. Postupak slanja disk komande odvija se u tri koraka:

1. otvaranje komandnog kanala 15 (naredbom **OPEN**)
2. saopštavanje komande disk jedinici (naredbom **PRINT#**)
3. zatvaranje komandnog kanala (naredbom **CLOSE**)

Sve ove naredbe mogu se zadati i u direktnom i u programskom načinu rada. Nazivi komandi mogu se zadavati i skraćeno korišćenjem samo prvog slova.

NEW — formatiranje diskete ili brisanje direktorijuma

Primer za ovu komandu se može videti u prethodnom opisu formatiranja. Sledeći primer prikazuje brisanje direktorijuma.

```
10 OPEN 15,8,15
20 PRINT#15,"N:IME DISKETE"
30 CLOSE 15
```

Postupak je isti kao kod formatiranja ali se ne navodi identifikacija id.

COPY — kopiranje bilo kog programa na istu disketu, pod drugim imenom

```
10 OPEN 15,8,15
20 PRINT#15,"C:IME NOVOG PROGRAMA=IME STAROG PROGRAMA"
30 CLOSE 15
```

Ukoliko je potrebno objediniti više programa u jedan, linija 20 treba da bude:

```
20 PRINT#15,"C:NOVI PROGRAM=STARI PROGRAM 1,STARI PROGRAM 2,..."
```

RENAME — promena imena programa na disketi

```
10 OPEN 15,8,15
20 PRINT#15,"R:NOVO IME PROGRAMA=STARO IME PROGRAMA"
30 CLOSE 15
```

SCRATCH — brisanje programa na disketi

```
10 OPEN 15,8,15
20 PRINT#15,"S:IME PROGRAMA 1,IME PROGRAMA 2,..."
30 CLOSE 15
```

U liniji 20 navedena su imena programa koje treba obrisati.

INITIALIZE — inicijalizacija

U rada sa diskom može doći do greške koja onemogućava dalji rad. Ova komanda vraća disk u stanje u kome je bio po uključenju.

```
10 OPEN 15,8,15
20 PRINT#15,"I"
30 CLOSE 15
```

VALIDATE — reorganizacija diska

Čestim upisivanjem i brisanjem programa i podataka na disketu pojaviće se neiskorišćeni prazni prostori, koji su međusobno odvojeni. Ti prostori su veličine dva do tri bloka tako da nije moguće u njih smestiti neke programe. Oni se mogu objediniti u jedinstveni, neprekidni prostor komandom **VALIDATE.**

```
10 OPEN 15,8,15
20 PRINT#15,"V"
30 CLOSE 15
```

Osim reorganizacije diskete, ova komanda se koristi za oslobađanje prostora koji zauzima neupotrebljiva nepravilno zatvorena datoteka. Ovakva datoteka je u direktorijumu označena znakom zvezdice (*), i ne može se obrisati komandom **SCRATCH.**

0 „PRIMERI	**„01**	**2A**
5 „PRIMER 1"		**PRG**
4 „PRIMER 2"		**PRG**
20 „PODACI"		*** SEQ**
635 BLOCKS FREE		

Izvršenjem komande **VALIDATE,** u prethodnom primeru oslobađa se 20 blokova koje zauzima datoteka „PODACI".

LOAD

Naredbom **LOAD** učitava se program iz spoljne jedinice u memoriju računara, a zatim eventualno startuje.

Spoljne jedinice (periferni uređaji) iz kojih se učitavaju programi skoro uvek su kasetofon ili disk jedinica. U radu sa kasetofonom program je zapisan na magnetnoj traci, tzv. kaseti. U radu sa disk jedinicom program je zapisan na magnetnom disku, tzv. disketi.

Opšti oblik naredbe za učitavanje je:

LOAD „ime", p, a

„ime" — je ime programa koji se učitava. Navodi se unutar navodnika. U radu sa kasetofonom ime nije obavezno, a za disk jedinicu jeste.

p — je broj spoljne jedinice. Za kasetofon je p=1, a za disk jedinicu je p=8. Navodi se, odvojeno zarezom, iza imena programa. Za kasetofon nije obavezno da se navede, što znači da će, ako se u naredbi izostavi broj spoljne jedinice, računar učitati program sa kasete.

a — je broj koji određuje adresu memorijske lokacije u računaru od koje će se učitavati program. Ako se ne navede, program će biti smešten u memoriji računara od lokacije 2048 ($0800). Ako je a=1 program će biti smešten od memorijske lokacije od koje je snimljen.

Izvršenjem naredbe za učitavanje zatvaraju se svi otvoreni kanali. Ako je naredba zadata direktno, pre nego što se izvrši učitavanje programa, izvrši se naredba **CLR,** tj. sve promenljive se postavljaju na nulu odnosno prazan string. Ako je naredba **LOAD** zadata programski, nakon učitavanja programa, izvršiće se autostart programa. Pri tome će biti sačuvane vrednosti svih promenljivih. Ovo omogućuje povezivanje više programa, pri čemu se prethodni program briše ali su sačuvane njegove promenljive.

Učitavanje programa sa kasetofona

Ispisivanjem naredbe **LOAD** i pritiskom tastera RETURN, računar ispisuje poruku:

PRESS PLAY ON TAPE (pritisnuti taster **PLAY** na kasetofonu)

Pritiskom tastera **PLAY** briše se sadržaj ekrana i otpočinje pretraživanje zapisa na traci. Kada se naiđe na prvi zapis, tj. program, kasetofon se zaustavlja i ispisuje se poruka:

SEARCHING
FOUND „ime programa"

Posle pauze od oko 10 sekundi otpočinje učitavanje nađenog programa. Pauza se može prekinuti pritiskom na jedan od sledećih tastera:

←, CTRL, C= ili SPACE.

Kada se ceo program učita, kasetofon se zaustavlja i ispisuje se poruka **READY.** Tada se program može startovati (naredba **RUN)** ili se može pristupiti njegovom modifikovanju (naredba **LIST).**

Naredba **LOAD** može se zadati i istovremenim pritiskom tastera SHIFT i RUN/STOP.

U okviru naredbe **LOAD** mogu se navesti, prema gore izloženom, ime programa koji se želi učitati, broj periferne jedinice sa koje se učitava program kao i adresa od koje će se smeštati program.

Primeri:	**LOAD**	Učitava se sledeći program sa trake.
	LOAD B$	Učitava se program čije je ime određeno promenljivom B$.
	LOAD ""	Učitava se sledeći program sa trake.
	LOAD "ime"	Učitava se program sa navedenim imenom.
	LOAD "ime", **1**, **1**	Učitava se program sa navedenim imenom, sa trake, i smešta se u memoriju, u adresni prostor iz kojeg je snimljen.

Učitavanje programa sa disk jedinice

Za učitavanje programa sa diska potrebno je zadati sledeću naredbu:

LOAD "ime programa", **8** (za bejzik program)
ili **LOAD** "ime programa", **8**, **1** (za mašinski program)

Nakon pritiska tastera RETURN ispisuju se poruke:

SEARCHING FOR „ime programa"
LOADING

Posle ovoga, po ispisivanju poruke **READY,** program je učitan.

Umesto imena programa moguće je navesti znak * čime će se učitati prvi program iz kataloga diska (direktorijuma). Takođe ime ne mora biti navedeno u potpunosti. Znakom * može se zameniti nedostajući deo imena.

Primeri:	**LOAD "*", 8**	Učitava se prvi program iz kataloga diska.
	LOAD "AB*", 8	Učitava se program čije ime počinje slovima **AB.**

Ukoliko ne postoji datoteka (engl. file) sa navedenim imenom, ili ako datoteka nije programska, prijaviće se izveštaj o grešci **?FILE NOT FOUND.**

SAVE

Naredbom **SAVE** snima se program na disk ili kasetu.

Izvršavanjem naredbe snima se program koji je u memoriji računara. Po završenom snimanju, program ostaje u memoriji nepromenjen.

Opšti oblik naredbe za snimanje je:

SAVE „ime programa", **p, a**

„ime" — je ime pod kojim se želi snimiti program. Za snimanje na kasetu ime nije obavezno, ali je preporučljivo navesti ga radi kasnijeg lakšeg nalaženja programa. Za snimanje programa na disketu ime je obavezno.

p — je broj spoljne jedinice. Za kasetofon je p=1, a za disk jedinicu je p=8. Navodi se, odvojeno zarezom, iza imena programa. Za kasetofon nije obavezno da se navede, što znači da će, ako se u naredbi izostavi broj spoljne jedinice, računar snimati program na kaseti.

a — je broj koji ima sledeća značenja:

a=1 Sačuvati početnu adresu programa što će omogućiti kasnije učitavanje programa od te adrese.

a=2 Stavljanje oznake za kraj trake iza programa (engl. end of tape marker).

a=3 Kombinacija a=1 i a=2.

Izvršenjem naredbe za snimanje programi se automatski snimaju na traku dva puta, radi provere pri učitavanju. Naredba **SAVE** može se zadati i programski.

Snimanje programa na kasetofonu

Ispisivanjem naredbe za snimanje i pritiskom na taster RETURN ispisuje se na ekranu poruka.

PRESS RECORD & PLAY ON TAPE (pritisnuti tastere RECORD i PLAY na kasetofonu)

Izvršivši to, sadržaj ekrana se briše i program se snima na kasetu. Po završenom snimanju kasetofon se zaustavlja i ispisuje se poruka **READY.**

Primeri:

SAVE	Program se snima na traci bez imena.
SAVE B$	Program se snima na traci pod imenom određenim promenljivom B$.

SAVE „MIKRO" Na traku se snima program pod imenom MIKRO.
SAVE „KNJIGA", 1, 3 Na traku se snima program KNJIGA sa početnom adresom programa i znakom za kraj programa na traci.

Snimanje programa na disk jedinici

Ispisivanjem naredbe za snimanje i pritiskom na taster RETURN ispisuju se na ekranu poruke:

SAVING ime programa
READY

Snimanjem bejzik ili mašinskih programa naredbama:

SAVE „ime programa", 8 (za bejzik program)
SAVE „ime programa", 8, 1 (za mašinski program)

Snimljena datoteka će biti programskog tipa (PRG).

Ukoliko treba učitati i izmeniti program, a zatim ga ponovo snimiti na disk pod istim imenom primenjuje se naredba:

SAVE „@: ime programa", 8

Pri tome se stara verzija briše.

VERIFY

Naredbom **VERIFY** upoređuje se snimak programa sa programom u memoriji računara.

Ova naredba koristi se odmah posle snimanja programa (naredba **SAVE**), da bi se izvršila provera ispravnosti snimka. Može se koristiti za proveru snimljenih programa kako na kaseti (traci) tako i na disketi. Može se zadati i direktno i programski.

Opšti oblik naredbe provere snimka je:

VERIFY „ime", p

„ime" — je ime snimljenog programa koji će se upoređivati sa programom u memoriji. U radu sa kasetofonom ime nije obavezno. Ako se izostavi za poređenje će biti upotrebljen prvi program na koji se naiđe na traci. U radu sa disk jedinicom ime je obavezno.

p — je broj spoljne jedinice. Za kasetofon je p=1, a za disk jedinicu je p=8. Ako se izostavi iz naredbe, provera će se izvršavati za snimak na traci.

U toku izvršavanja naredbe **VERIFY** ako se pojavi bilo kakvo odstupanje u snimku programa i programa u memoriji računara, prijaviće se izveštaj o grešci **?VERIFY ERROR.** Kod rada sa mašinskim programima izveštaj o grešci može se javiti i onda kada je program dobro snimljen, ukoliko se nalazi „iza" bejzik ili Kernal ROM-a (videti poglavlje 8).

Pri radu sa kasetofonom nakon ispisivanja naredbe provere zapisa i pritiskom na taster RETURN ispisuje se poruka:

PRESS PLAY ON TAPE (pritisnuti taster PLAY na kasetofonu)

Izvršivši to, uz prethodno premotanu traku ispred početka snimka programa, briše se sadržaj ekrana. Nailaskom na snimak programa ispisuju se poruke:

SEARCHING
FOUND ime programa

Nastavljanjem i završetkom provere ispisuju se poruke **VERIFYING** i **OK.**

Pri radu sa diskom nakon ispisivanja naredbe provere snimka i pritiskom na taster RETURN ispisuju se poruke:

SEARCHING FOR ime programa
VERIFYING
OK
READY
Ime programa ne mora biti navedeno u potpunosti. Znakom * može se zameniti nedostajući deo imena.

Primeri: **VERIFY „*", 8**
VERIFY „AB*", 8, 1

U prvom primeru sadržaj memorije se upoređuje sa prvim programom iz direktorijuma diska. U drugom primeru sadržaj memorije se upoređuje sa programom čije ime počinje slovima AB.

4.2.11 Datoteke

Svaki program radi sa nekim podacima. Ukoliko postoji velika količina podataka oni mogu biti organizovani u složenije strukture, datoteke (engl. file). Svaka datoteka predstavlja niz osnovnih elemenata logično povezanih u celinu. Ti elementi zovu se slogovi (engl. records). Svaki slog može se sastojati od jednog elementarnog podatka tj. broja ili stringa. Unutar sloga može biti i više elementarnih podataka različitog tipa, od kojih svaki čini polje (engl. field). Odgovarajuća polja u različitim slogovima sadrže isti tip elementarnog podatka.

Primer: Datoteka adresa i brojeva telefona;

Jedan slog može imati sledeća polja: ime, prezime, ulica i broj, grad, broj telefona. Svakoj osobi pridružuje se jedan slog, što se može prikazati na sledeći način:

	polje 1	polje 2	polje 3	polje 4	polje 5
Osoba 1 (1. slog)	Petar	Petrović	7. Jula 96	Beograd	186 – 453
Osoba 2 (2. slog)	Jovan	Jovanović	Končareva 3	Zagreb	436 – 119
–	–	–	–	–	–
–	–	–	–	–	–
–	–	–	–	–	–

Podaci mogu biti organizovani u tri tipa datoteka:
1. Sekvencijalne (engl. sequential)
2. Direktne (engl. random)
3. Relativne (engl. relative)

Sekvencijalne datoteke

Sekvencijalne datoteke predstavljaju niz podataka koji se čuvaju u istom redosledu u kome su uneti. Prvi podatak koji je upisan u datoteku biće i prvi podatak koji će biti pročitan. Ako je potrebno izmeniti neki podatak u datoteci, potrebno je učitati celu datoteku pa je, po obavljenoj izmeni, ponovo upisati. Kraj datoteke je obeležen specijalnim znakom EOF (engl. end of file marker). Novi podatak se uvek dopisuje na kraju datoteke, a EOF

se pomera za jedno mesto dalje. Svakom podatku se pristupa preko jednog rezervisanog dela memorije (engl. buffer) koji ima ulogu prozora kroz koji se vide elementarni podaci. Koji će se podatak videti kroz ovaj prozor, zavisi od sadržaja jedne promenljive zvane pokazivač (engl. pointer). Kako se datoteka čita, pokazivač se pomera ka sledećem podatku sve do kraja, tj. nailaska na EOF. Pokazivač se može svakog trenutka vratiti na početak datoteke posebnom naredbom (engl. reset).

Sekvencijalne datoteke mogu biti organizovane i na traci i na disku. Koriste se za podatke koji ne moraju često da se menjaju, a pogodne su u tome što na disku ne zauzimaju mnogo prostora. Upotrebom kasetofona i trake mogu se koristiti samo sekvencijalne datoteke.

Jedan poseban vid organizacije sekvencijalnih datoteka je bejzik naredba **DATA**. Ova datoteka se čita naredbom **READ**, prozor (bafer) je promenljiva kojoj se dodeljuje vrednost pri čitanju, a reset funkciju obavlja naredba **RESTORE**.

Direktne datoteke

Kod direktnih datoteka, svakom podatku se može pristupiti direktno, bez prethodnog čitanja drugih podataka. Ovakav tip datoteke se može organizovati samo korišćenjem diska, jer se pri tome zna gde se koji podatak nalazi. Tom podatku se zatim pristupa direktno pomeranjem glave za čitanje/upisivanje na odgovarajuću stazu i odgovarajući blok. Pristup podacima se vrši preko jednog od četiri bafera (veličine 256 bajtova) koji se nalaze u samoj disk jedinici. Pre nego što se počne sa upisivanjem u direktnu datoteku, ona mora da se kreira. To se obavlja definisanjem polja i slogova koji će se koristiti. Takođe mora postojati i neko ključno polje po kome će se datoteka pretraživati, **npr. praznine u datoteci adresa i brojeva** telefona. Na osnovu ovog ključnog polja, formira se sekvencijalna datoteka informacija o stazi i bloku gde se nalazi odgovarajući slog. Čitanje se obavlja na sledeći način: učita se prvo sekvencijalna datoteka ključnih polja, pa se na osnovu zadate ključne reči (npr. prezime neke osobe) pronađe staza i blok gde se na disku nalazi ceo slog. Zatim se preko bafera ceo blok učita u memoriju, pa se iz njega izdvaja potreban slog (npr. osoba 2), a zatim iz sloga potrebno polje (npr. broj telefona).

Relativne datoteke

Relativne datoteke su direktne datoteke kod kojih se ne vodi računa o fizičkom položaju slogova na disku. Tu brigu preuzima DOS kreirajući poseban direktorijum relativne datoteke sa informacijama o tome gde se nalaze svi slogovi datoteke. Taj direktorijum čine tzv. sektori slogova (engl. side sector) i oni oslobađaju korisnika kreiranja posebne sekvencijalne datoteke sa indeksima ključnih polja. Za razliku od standardne direktne datoteke gde se učitava ceo blok, kod relativne datoteke se učitava samo željeni slog.

Programske datoteke

Programske datoteke nisu datoteke u pravom smislu, već način pamćenja programa na disku. Mogu biti dva tipa:

1. PRG — za programe rezidentne u Komodorovoj memoriji. Ovo može biti bilo koji programski tekst ili mašinski program sačuvan pomoću naredbe **SAVE** i može biti učitan pomoću naredbe **LOAD.**

2. USR — za mašinske programe rezidentne u memoriji kontrolera disk jedinice.

OPEN

Ovom naredbom otvara se kanal za ulaz i/ili izlaz na periferni uređaj ili datoteku (engl. file).

Opšti oblik ove naredbe je:

OPEN d,p,a, „ime, tip, mod"

d — logički broj datoteke. Potrebno je dodeliti ga datoteci. Sve naredbe za rad sa datotekama pozivaju se na njega. Može biti u opsegu od 0 do 255, ali preporučuje se korišćenje u opsegu od 0 do 127.

p — broj perifernog uređaja. Svaki periferni uređaj tj. spoljna jedinica (štampač, kasetofon, disk jedinica, ploter...) ima svoj broj. Ti brojevi su:

0 — tastatura
1 — kasetofon
2 — RS-232
3 — ekran
4 — štampač
5 — štampač
6 — ploter
7 —
8 — disk jedinica

Ako se broj perifernog uređaja izostavi, Komodor će podrazumevati razmenu podataka sa kasetofonom.

a — sekundarna adresa perifernog uređaja. Dodatno određuje rad perifernog uređaja. Kod rada sa kasetofonom vrednost 0 sekundarne adrese određuje učitavanje sa kasetofona, vrednost i upisivanje na traku, a vrednost 2 određuje postavljanje EOT znaka (end of tape marker), na kraju datoteke prilikom njenog zatvaranja. EOT znak onemogućava očitavanje podataka van datoteke što bi dovelo do izveštaja o grešci **?DEVICE NOT PRESENT.** Ako se sekundarna adresa izostavi podrazumevaće se da se radi o učitavanju sa trake. U radu sa disk jedinicom sekundarna adresa mora se navesti. Vrednosti od 2 do 14 su za rad sa podacima, a ostale imaju posebno značenje za DOS (disk operativni sistem). Vrednosti 0 i 1 rezervisane su za rad sa kanalima pri snimanju i čitanju sa diska. Vrednost 15 predviđena je za tzv. komandni kanal ili kanal greške.

ime — string dužine od 1 do 16 karaktera koji se dodeljuje datoteci kao njeno ime. U radu sa štampačem ili kasetofonom nije obavezno.

tip — vrsta datoteke. Datoteke mogu biti:

PRG — programska
SEQ — sekvencijalna
USR — korisnička
REL — relativna (sa direktnim pristupom)

mod — označava način pristupa datoteci.

R — očitavanje
W — upisivanje

Naredba OPEN mora biti izvršena pre ostalih ulazno izlaznih naredbi. U protivnom prijaviće se izveštaj o grešci **?FILE NOT OPEN.** Ako se otvara datoteka (tačnije rečeno

kanal) za očitavanje koja ne postoji, prijaviće se izveštaj o grešci **?FILE NOT FOUND**. Ako se otvara datoteka za upisivanje koja već postoji, u radu sa diskom, prijaviće se izveštaj o grešci **?FILE EXISTS**. Konačno, ako se otvara već otvoreni kanal, prijaviće se izveštaj **FILE OPEN**.

Primeri:

10 OPEN 1,4	(izlaz na štampač)
10 OPEN 1,2,0 CHR$(6)	(RS 232 kanal)
10 OPEN 15,8,15	(komandni kanal diska)
10 OPEN 2,3	(izlaz na ekran)

CLOSE

Ovom naredbom zatvara se kanal ka perifernom uređaju ili datoteci.

Iza reči **CLOSE** treba navesti logički broj datoteke koja se zatvara. U radu sa kasetofonom ili diskom obavezno je izvršenje ove naredbe. U protivnom može doći do nepovratnog gubljenja podataka. U radu sa ostalim ulazno izlaznim uređajima nije obavezno izvršenje ove naredbe, ali je poželjno jer oslobađa memoriju za druge datoteke.

Primer: `10 CLOSE 5`

Navedenim primerom zatvara se kanal, a time i datoteka označena brojem 5, koji je prethodno morao biti dodeljen naredbom **OPEN**.

GET#

Ova naredba omogućava očitavanje jednog po jednog karaktera iz ulazno izlaznih uređaja, odnosno datoteka.

Opšti oblik ove naredbe je:

GET# d, v1, v2, v3,...

d – broj prethodno otvorene datoteke iz koje se vrši očitavanje.

v1, v2, v3... – promenljive, međusobno odvojene zarezima, kojima se dodeljuju očitane vrednosti. Mogu biti i brojne i string.

Naredba **GET#** odgovara naredbi **GET**, s tim što podaci ne dolaze sa tastature već iz naznačene datoteke. Ako se ne učita ni jedan karakter promenljivoj se dodeljuje vrednost 0, ako je brojna promenljiva, odnosno dodeljuje joj se prazan string ako je string promenljiva.

Kontrolni karakteri načina ispisivanja i pomeranja kursora tretiraju se podjednako kao bilo koji karakter.

Najčešća primena ove naredbe je u očitavanju sa trake ili diskete. Moguće ju je koristiti i u očitavanju jednog po jednog karaktera sa ekrana. Pri tome se kursor, koji ukazuje na mesto očitavanja, pomera nakon svakog čitanja za jedno mesto udesno. Na kraju svake linije (reda) ekrana očitava se karakter sa kodom 13 (RETURN).

Primer:

```
10 PRINT "(CLR)"
20 PRINT "ASOFGHJKL"
30 PRINT "(HOME)"
40 OPEN 1,3
50 FOR N-1 TO 10
60 GET #1,A$(N)
70 NEXT N
80 CLOSE 1
90 PRINT:FOR N-1 TO 10
100 PRINT A$(N)
110 NEXT N
```

Linijama 10 i 20 briše se sadržaj ekrana i u drugom redu ispisuje se string ASDFGHJKL. Komandom HOME (taster CLRHOME), linija 30 vraća kursor na poziciju gornjeg levog karaktera ekrana. Nakon otvaranja kanala na ekranu, linija 40, u petlji se očitavaju karakteri ispisani na ekranu: linije 50, 60 i 70. Treba uočiti da je očitavanje otpočelo u redu ispod reda u kome se nalazio kursor.

Nakon zatvaranja kanala i prenošenja kursora, tj. ispisivanja, očitani karakteri se ispisuju red niže.

INPUT#

Ovom naredbom očitavaju se podaci iz perifernog uređaja, i dodeljuju naznačenim programskim promenljivama.

Podatak koji se očitava može biti u dužini do 80 karaktera, za razliku od naredbe **GET#**, gde je u dužini jednog karaktera. Opšti oblik naredbe je:

INPUT# d, v1, v2, v3,...

d — broj prethodno otvorene datoteke iz koje se vrši očitavanje.

v1, v2, v3,... — promenljive, međusobno odvojene zarezima, kojima se dodeljuju očitane vrednosti. Mogu biti brojne i string.

Pri očitavanju promenljive, naredbom **INPUT#**, pod njenim krajem podrazumevaju se separatori (engl. delimiter) RETURN (kôd 13), zarez **(,)**, tačka zarez **(;)** ili dvotačka **(:)**. Ako se želi uključenje i separatora u podatke, treba ih navesti unutar navodnika (npr. „,").

U slučaju da se brojnoj promenljivoj dodeljuju podaci koji nisu brojni, prijaviće se izveštaj o grešci **?BAD DATA.** U slučaju da je podatak veće dužine od 80 karaktera, prijaviće se izveštaj o grešci **STRING TOO LONG.**

Pri očitavanju sa ekrana (periferni uređaj 3), očitaće se ceo fizički red (40 karaktera), a zatim će se kursor pomeriti jedan red niže.

Primer:

```
10 REM CITANJE SEQ. DATOTEKE
20 INPUT "BROJ SLOGOVA";N
30 DIM A$(N),B$(N),C$(N),D$(N),E$(N)
40 PRINT CHR$(147)
50 OPEN 5,8,5,"PRIMER DATOTEKE,S,R"
60 PRINT "PREZIME"TAB(13)"IME"TAB(21)"MESTO"TAB(29)"BROJ"
70 PRINT"--------------------------------------"
80 FOR I=1 TO N
90 INPUT#5,A$(I),B$(I),C$(I),D$(I),E$(I)
100 PRINTA$(I)TAB(13)B$(I)TAB(21)C$(I)TAB(29)E$(I)
110 NEXT I
120 CLOSE 5
```

PRINT#

Ova naredba koristi se za upisivanje podataka u datoteku.

Primenjuje se u sledećem obliku. Iza reči **PRINT#** navodi se broj datoteke, a iza njega odvojeno zarezom, navode se promenljive čije se vrednosti upisuju u datoteku. Promenljive moraju biti odvojene znacima interpunkcije kao separatorima (zarezom ili tačka zarezom). Umesto promenljivih mogu se navesti i izrazi, a takođe i konkretne vrednosti, brojne ili string.

Upotreba separatora, zareza i tačka zareza, pri upisivanju u datoteke ima drugačije dejstvo u odnosu na njihovu upotrebu u naredbi ispisivanja na ekran **PRINT.** Kada se naredba **PRINT#** koristi za upisivanje upotrebom zareza vrednosti koje se upisuju razdvajaju se sa 10 karaktera praznog polja. Upotrebom tačka zareza između vrednosti izvršiće se njihovo upisivanje kao kontinualni niz karaktera. Brojnim podacima prethodi prazno

polje (u slučaju negativne vrednosti prethodi znak minus), a takođe praćeni su praznim poljem.

U slučaju da se lista podataka, koja se upisuje, ne završava zarezom ili tačka zarezom, dolazi do upisivanja kontrolnog karaktera **RETURN** (kod 13). Ako je lista završena zarezom ili tačka zarezom, ne upisuju se dodatni kodovi. Bez obzira na separatore kojim se završavaju podaci, sledećom upotrebom naredbe **PRINT#**, upisivanje počinje od prve sledeće pozicije za karakter.

Prilikom upisivanja podataka separatori: zarez, tačka zarez i **RETURN**, mogu se zadati u obliku karaktera njihovog koda (**CHR$(44)** za zarez, **CHR$(59)** za tačka zarez i **CHR$(13)** za RETURN). Pogodno ih je definisati kao string promenljive i kao takve koristiti.

Primer:

```
10 REM UPISIVANJE U SEQ. OATOTEKU
20 INPUT "BROJ SLOGOVA";N
30 OIM A$(N),B$(N),C$(N),D$(N),E$(N)
40 OPEN 5,8,5,"PRIMER OATOTEKE,S,W"
50 FOR I=1 TO N
60 PRINT "(";I;")"
70 INPUT "PREZIME";A$(I)
80 INPUT "IME";B$(I)
90 INPUT "OATUM ROOJENJA";O$(I)
100 INPUT "MESTO ROOJENJA";C$(I)
110 INPUT "REGISTARSKI BROJ";E$(I)
120 R$=CHR$(13)
130 PRINT#5,A$(I)R$B$(I)R$C$(I)R$O$(I)R$E$(I)
140 NEXT I
150 CLOSE 5
```

U cilju pravilnog učitavanja podataka naredbom **INPUT#**, podatke je potrebno odvojiti sa **CHR$(13)**, koji označava kraj podatka.

CMD

Ovom naredbom obavlja se prelazak sa osnovnog ulazno izlaznog uređaja (ekran) na naznačeni ulazno izlazni uređaj.

CMD d, string

d — broj datoteke (kanala). Datoteka može biti na disku, traci, štampaču ili nekom drugom perifernom uređaju.

string — bilo koji string. Ne mora da se navede. Ako se navede šalje se na naznačeni uređaj.

Izvršenjem naredbe **CMD** se **PRINT** i **LIST** naredbe neće izvršavati ispisivanje na ekranu, već na naznačenom uređaju. Zbog toga je upotreba naredbe CMD uobičajna u radu sa štampačem.

Sledeća naredba ispisuje (lista) tekst programa na štampaču:

Primer: **OPEN 1,4:CMD1,„IME PROGRAMA":LIST**

Za vraćanje ispisivanja na ekran potrebno je ispisati jedan prazan red (**PRINT# 1** za datoteku označenu brojem 1), a zatim izvršiti zatvaranje datoteke (**CLOSE 1** za datoteku 1).

Pojava bilo koje greške vratiće ispisivanje na ekran, ali potrebno je naznačenom uređaju poslati komandu za ispisivanje jednog praznog reda.

STATUS

Naredba **STATUS** očitava stanje poslednje obavljene ulazno izlazne operacije nad otvorenom datotekom. Status, odnosno stanje može se očitati sa bilo kog ulazno izlaznog uređaja i izražava se brojnom vrednošću.

Data je tabela sa status vrednostima za kasetofon, štampač, disk i serijski (IEC) interfejs.

težina STATUS bita	STATUS vrednost	učitavanje sa trake	IEC	VERIFY + LOAD (traka)
0	1		Pri upisu ispad iz sinhronizacije	
1	2		pri čitanju ispad iz sinhronizacije	
2	4	kratki blok		kratki blok
3	8	dugi blok		dugi blok
4	16	greška u učitavanju		bilo koje ne poklapanje
5	32	greška u zbiru provere (checksum)		greška u zbiru provere
	64	EOF kraj datoteke	EOI (end or identify)	
	-128	EOT kraj trake	ne postoji periferija	EOT kraj trake

Primer:

```
10 REM ŠTAMPANJE SEKVENCIJALNE DATOTEKE
20 OPEN 2,8,4,"DATOTEKA,SEQ,R"
30 OPEN 1,4
40 INPUT#2,A$
50 IF ST=64 THEN 90
70 PRINT#1,A$
80 GOTO 40
90 CLOSE 1:CLOSE 2
```

4.3 IZVEŠTAJI

Izveštaji su poruke kojima računar obaveštava korisnika o prestanku izvršavanja bejzik programa. U izveštaju se daje kratak opis uzroka zaustavljanja i broj programske linije u kojoj je došlo do prekida. Ako je do prekida došlo, ne u programskom već u direktnom načinu rada, broj linije će biti izostavljen.

Primer: **?BAD DATA ERROR IN 100**

Do prekida u izvršavanju programa došlo je u liniji 100 zbog pogrešnih podataka.

U daljem tekstu navedeni su izveštaji i okolnosti pod kojima se javljaju.

BAD DATA (pogrešan podatak)

U programu se očekuje brojna vrednost, a očitana je string vrednost.

BAD SUBSCRIPT (pogrešan indeks)

Indeks je veći od dimenzije promenljive ili je u indeksu pogrešan broj.

BREAK (prekid)

Izvršavanje programa je prekinuto usled izvršenja naredbe **STOP** ili zbog pritiska na taster **STOP.**

CAN'T CONTINUE (ne može se nastaviti)

Nije moguće nastaviti izvršavanje programa naredbom **CONT** iz sledećih razloga: program nije startovan (naredba **RUN**), došlo je do pojave greške u izvršavanju programa ili se obavljalo uređivanje teksta programa (editovanje).

DEVICE NOT PRESENT (ne postoji periferna jedinica)

Odgovarajuća ulazno izlazna jedinica ne prima odgovarajuće (ulazno izlazne) naredbe.

DIVISION BY ZERO (deljenje sa nulom)

Deljenje sa nulom nije dozvoljeno.

EXTRA IGNORED (višak odbačen)

Ukoliko se pomoću naredbe **INPUT** unese više podataka nego što se očekuje, višak će biti odbačen i prijaviće se ovaj izveštaj.

FILE NOT FOUND (nije nađena datoteka)

Ako se na traci traži određena datoteka i ako se ona ne nađe već se naiđe na znak za kraj trake (EOT) prijaviće se ovaj izveštaj. U radu sa diskom ako se na disketi ne nađe tražena datoteka prijaviće se ovaj izveštaj.

FILE NOT OPEN (datoteka nije otvorena)

Ako prethodno nije izvršena naredba **OPEN,** a izvršavaju se naredbe **GET#, INPUT#, PRINT#** ili **CMD,** prijaviće se ovaj izveštaj.

FILE OPEN (otvorena datoteka)

Pokušano je otvaranje već otvorene datoteke (upotrebljen je isti broj datoteke).

FORMULA TOO COMPLEX (izraz previše složen)

Izraz ili ima previše zagrada ili se mora razdvojiti u dva ili više manjih izraza.

ILLEGAL DIRECT (zabranjeno u direktnom načinu)

Direktnim zadavanjem naredbi koje se mogu izvršavati samo programski, prijavljuje se ovaj izveštaj.

ILLEGAL QUANTITY (nedozvoljena vrednost)

Broj koji je upotrebljen kao argument u naredbi nalazi se van dozvoljenog opsega.

LOAD ERROR (greška u učitavanju)

Pojavila se greška u učitavanju sa trake. Videti: Rad sa kasetofonom.

NEXT WITHOUT FOR (NEXT bez **FOR)**

U izvršavanju programa naišlo se na naredbu **NEXT** kojoj nije prethodila naredba **FOR...**, ili promenljiva (indeks) u naredbi **NEXT** ne odgovara promenljivoj u naredbi **FOR.**

NOT INPUT FILE (nije datoteka za čitanje)

Do pojave izveštaja je došlo jer je pokušano očitavanje iz datoteke koja je naznačena kao datoteka samo za upisivanje.

NOT OUTPUT FILE (nije datoteka za upis)

Do pojave izveštaja je došlo jer je pokušano upisivanje podataka u datoteku koja je naznačena kao datoteka samo za očitavanje.

OUT OF DATA (nema podataka)

Izvršenjem naredbe **READ**, nisu nađeni podaci u naredbi **DATA**. Ovaj izveštaj se pojavljuje i prilikom uređivanja teksta programa.

OUT OF MEMORY (nema slobodne memorije)

Program je postao dug tako da u memoriji računara nema više mesta za dalje proširenje programa i za nove promenljive. Moguće je i da je upotrebljeno previše **FOR** petlji, **GOSUB** naredbi ili je rezervisan preveliki prostor naredbom **DIM**.

OVERLOW (premašenje)

Rezultat izračunavanja je veći od najvećeg dozvoljenog broja, a to je 1.70141884E38.

REDIM'D ARRAY (ponovljeno dimenzionisanje)

Višedimenzionalne promenljive (vektor ili matrica) definišu se samo jedanput tokom izvršavanja programa. Ako se višedimenzionalna promenljiva upotrebi pre izvršenja naredbe DIM izvršiće se automatski definisanje promenljivih do indeksa 10. Zbog toga ako se ponovo pokuša definisanje prijaviće se ovaj izveštaj.

REDO FROM START (ponovo)

U izvršavanju naredbe **INPUT** unesene su string vrednosti, a ne brojne vrednosti koje se očekuju. Izveštaj obaveštava da je potrebno uneti ispravne vrednosti.

RETURN WITHOUT GOSUB (RETURN bez **GOSUB)**

Naišlo se na naredbu **RETURN**, a pre toga u programu nije izvršena odgovarajuća **GOSUB** naredba.

STRING TOO LONG (string previše dug)

Pojavio se string koji je duži od 255 karaktera.

SYNTAX ERROR (greška u sintaksi)

U ispisivanju programa moraju se koristiti samo za to dozvoljene reči. Takođe moraju se poštovati pravila pisanja naredbi i programski linija.

TYPE MISMATCH (greška u tipu podatka).

Do prijavljivanja izveštaja će doći ako se umesto broja upotrebi string i obrnuto.

UNDEF'D FUNCTION (nedefinisana funkcija)

Naišlo se na naredbu **FN** ali nije nađena njena definicija.

UNDEF'D STATEMENT (nedefinisana linija)

Naredbom **GOTO, GOSUB** ili **RUN** pokušalo se izvršavanje programske linije koja ne postoji.

VERIFY ERROR (greška u proveri)

Zapis programa na traci ili disku nije isti sa programom u memoriji.

5
Principi programiranja

Namena ovog poglavlja je da upozna čitaoca sa materijom koja je neophodna za korektnu i efikasnu izradu programa. Izloženo pokriva tradicionalno programiranje i najznačajniji savremeni način programiranja — struktuirano programiranje. Pri tome nije izvršeno ograničavanje na određeni programski jezik, ali je, tamo gde je potrebno, ukazano na specifičnosti Komodorovog standardnog bejzika i Sajmons bejzika.

5.1 OSNOVNI POJMOVI

Sposobnost računara da obrađuje i pamti veliku količinu podataka omogućuje njihovu primenu u rešavanju velikog broja problema. To su problemi čije se rešavanje sastoji u određenom broju jasno definisanih radnji koje je potrebno uraditi sa podacima po unapred određenom redosledu. Praktično radi se o rutinskim poslovima. To ne mogu biti kreativni poslovi. Od računara se ne može očekivati da iznalazi nove puteve rešavanja, niti da samostalno rešava postavljene zadatke. Svrha računara je da preuzme rutinske poslove, a da čoveku ostavi više vremena za kreativno delovanje.

Pretpostavimo da treba rešiti sledeći problem:

Izračunati zbir prvih n prirodnih brojeva. (A)

Da bi se ovaj problem rešio neophodno je poznavanje osnovnih elemenata aritmetike. Potrebno je znati da su n prvih prirodnih brojeva sledeći brojevi: 1, 2, 3,..., n−1, n (npr. ako je n jednako 6 to su brojevi: 1, 2, 3, 4, 5 i 6).

Rešavanje problema zahteva njegovo raščlanjivanje na izvestan broj jednostavnijih potproblema. Takođe je potrebno utvrđivanje redosleda njihovog rešavanja. To može biti na primer sledeće rešavanje:

1. Neka z bude 0.
2. Neka a bude 1.
3. Izračunati z+a.
4. neka z bude z+a. (B)
5. Ako je a=n rešenje je z.
6. Povećaj a za 1.
7. Ponavljaj postupak počevši od tačke 3.

Gornjih sedam tačaka (B) su opis postupaka koji daju zbir prvih n prirodnih brojeva (podrazumeva se da je n unapred zadata vrednost). Rezultat je vrednost promenljive z po završetku postupka.

Navedeni postupak se sastoji od određenog broja radnji koje je potrebno obaviti u određenom redosledu i naziva se algoritam. Uočava se da se u opisu postupaka rešavanja,

sem radnji koje je potrebno obaviti, nalaze i podaci nad kojima se te radnje obavljaju. U ovom slučaju podaci su brojne vrednosti i označene su slovima n, z i a. Tokom izvršavanja postupka vrednosti označene slovima se menjaju i zbog toga se nazivaju promenljive.

Raščlanjivanje problema (A) na postupak (B) potrebno je izvršiti zato što zasada ne postoji računar koji bi samo na osnovu datog problema, pod (A), mogao naći rešenje. Opis postupaka za rešavanje problema, pod (B), i dalje je neprihvatljiv za računar. Potrebno je izvršiti dalje raščlanjivanje agoritma (B) sve dok se algoritam na kraju ne izrazi programskim jezikom, tj. dok se ne prevede u formu programa.

Poznavanjem nekog od programskih jezika može se pristupiti prevođenju, za dati primer, algoritma (B) u oblik izražen programskim jezikom. Navedeni algoritam još uvek nije pogodan za prevođenje u program. Nije određeno unošenje zadatog podatka n. Nakon daljeg preciziranja algoritma u tom pogledu može se izvršiti njegovo izražavanje programskim jezikom bejzik. Jedno od mogućih rešenja je:

```
10 INPUT N
20 LET Z=0
30 LET A=1
40 LET Z=Z+A
50 IF A=N THEN PRINT Z:STOP                (C)
60 LET A=A+1
70 GOTO 40
```

Na osnovu do sada izloženog zaključuje se da rešavanje zadataka uz pomoć računara započinje analizom problema. Nastavlja se raščlanjavanjem problema i preciznijem opisivanjem radnji koje je potrebno obaviti da bi se stiglo do rešenja. Formira se algoritam, od uopštenog do potpuno preciziranog, koji se zatim prevodi u program. To prevođenje u program, fizičko pisanje programa, zahteva poznavanje programskog jezika i iskustvo u njegovoj primeni. Pri tome mogućnosti pojave grešaka su velike, ali svakako najteži i najznačajniji deo u dobijanju programa je formiranje algoritma.

Formiranje algoritma je iznalaženje rešenja problema. Ono zahteva veoma dobro poznavanje problematike koja se rešava. U datom primeru, a naročito u složenijim problemima prelaz od (A) do (B) nije jednostavan. Zahteva određeno logičko rasuđivanje, prethodna znanja i iskustva koja nisu u direktnoj vezi sa poznavanjem rada računara. Nalaženjem dovoljno preciznog algoritma, njegovo izražavanje u obliku programa (C) predstavlja manji problem, i može se čak okvalifikovati kao rutinski posao.

Po formiranju programa sledi njegovo testiranje i izvođenje koje treba da potvrdi postavljene ciljeve, a takođe i da omogući eventualne ispravke programa.

Konačno poslednji ali ništa manje značajan korak je kompletiranje dokumentacije. Dokumentacija predstavlja nezamenljiv i neophodan deo svakog programa. Potrebno je jasno i pregledno od početka do kraja razvoja programa voditi evidenciju o svim aktivnostima na razvoju programa. U protivnom rad uložen u program će najverovatnije biti nepovratno izgubljen.

5.2 RAZVOJ PROGRAMA

Analiza problema

Cilj analize problema je detaljno sagledavanje svega onoga što je značajno za rešavanje problema koji se želi obrađivati računarom. Neophodno je jasno sagledati ceo proces za

koji se želi napisati program. Bilo da je to neki proračun, obrada podataka, animacija, simulacija, igra ili nešto drugo, potrebno je tačno utvrditi šta se želi postići. Pri tome se ne treba vezivati za način rada računara ili način kako će to biti postignuto. Svakako, treba imati na umu mogućnosti računara na kome će se program realizovati.

U ovoj fazi pre svega potrebno je jasno definisati koje se informacije žele dobiti, odnosno šta se želi ostvariti programom sa gledišta korišćenja gotovog programa. Pri tome treba odrediti i koji su podaci za to potrebni.

Takođe u ovoj fazi treba odrediti i globalni postupak koji će od zadatih podataka omogućiti odbijanje željenih informacija. Taj postupak treba da je na najopštijem nivou, sa ciljem da ukaže na postojanje rešenja postavljenog zadatka. Postupak može biti izabran iz nekoliko određenjih postupaka. Kriterijum izbora pri tome može biti vrlo raznolik: od vremena i aktivnosti koje se postavljaju pred realizaciju programa, preko funkcionalnosti programa, ograničenja računara, do estetskih razloga. Izbor između utvrđenih postupaka ostavlja slobodan prostor programeru za njegovo iskustvo i kreativnost.

Algoritam

Program u svome radu vrši obradu određenih podataka. Podaci se u toku izvršavanja programa uzimaju iz spoljne sredine ili memorije računara, vrši se njihova obrada, a dobijeni rezultati se pamte ili prezentiraju na nekom od izloženih uređaja.

U ovoj, drugoj fazi, potrebno je jasno definisati postupke koje treba obaviti nad podacima. Pri tome svaki korak mora biti precizno određen tako da dobijeni rezultat ne zavisi od osobe koja je program upotrebila, od toga po koji se put program izvršava ili od raznih drugih uslova koji ne smeju uticati na rad programa. Takođe je potrebno sagledati da li postoje podaci za koje se ne mogu izračunati rešenja (u prethodnom primeru to mogu biti negativne vrednosti za n). Treba predvideti šta će program tada izvršavati. Isto važi i za neodgovarajuću upotrebu programa.

Opis postupaka za nalaženje rešenja, kao što je u prethodnoj tački rečeno, naziva se algoritam. Prve verzije algoritma treba da budu najopštijeg tipa. U njima su opšti podaci i izrazi koji operišu nad tim podacima. U prvoj fazi razvoja programa algoritam je na apstraktnom nivou pa se naziva i apstraktnim algoritmom.

Rešavanje treba da je nezavisno od računara koji će se koristiti. Algoritam treba izraziti jezikom najvišeg izražajnog nivoa — govornim jezikom. Time se najsigurnije postiže formiranje ispravnog algoritma. Problemi koji se pri tome mogu rešavati su najopštijeg nivoa, to je problematika samog procesa za koji se želi napisati program.

Rasčlanjivanje algoritma, preciznijim opisivanjem podataka i radnji koje je potrebno obaviti predstavlja dalje aktivnosti ka formiranju programa. Upravo taj proces dobijanja preciznog algoritma čini aktivnost koja se naziva programiranje.

Konačno, detaljni algoritam se izražava programskim jezikom, tj. ispisuje se program. To je potrebno uraditi tek u ovoj fazi, pošto su prethodno donesene sve odluke značajne za dobijanje željenih rezultata. Algoritam se prevodi u program korišćenjem pravila upotrebe programskog jezika: Kako će to biti urađeno zavisi od mogućnosti jezika, a i od znanja i iskustva programera.

Poznavanje programskog jezika je neophodno za kvalitetno pisanje programa. Njegovim poznavanjem moguć je izbor odgovarajućih naredbi, grupa naredbi ili struktura kojima će se dobiti korektno i efikasno rešenje.

U toku pisanja programa potrebno je obratiti pažnju na sledeće:

Ne sme se dozvoliti da neke vrednosti podataka dovedu do neregularnog rada programa (npr. deljenje sa nulom).

Program treba pisati jasno i pregledno, ne samo što se tiče dokumentacije, već i što se tiče formiranja programskih linija. To je neophodno ne samo za efikasnu, već i za tačnu izradu programa. Program treba napisati jasno, tako da bude razumljiv svakom drugom poznavaocu programiranja. U protivnom, nakon izvesnog vremena program će biti nerazumljiv i samom autoru programa.

Pred program se postavljaju dva zahteva. Jedan je da se program što brže izvršava, a drugi je da program zauzima što manje mesta u memoriji računara, tj. da bude što kraći. Ova dva zahteva su po pravilu suprotna. U praksi važniji zahtev će odrediti izbor rešenja.

U izborima realizacije dela programa ili celog programa prednost treba davati rešenjima opštijeg tipa. Treba izbegavati rešenja koja ne koriste standardne programske strukture, iako to mogu biti rešenja koja se odlikuju većom brzinom izvršavanja ili manjom potrošnjom memorije.

U pisanju programa treba imati u vidu delove programa koji su nezavisni od tipa računara i delove koji su zavisni. Pažljivim izborom koji će se deo kako rešiti omogućuje se prenosivost programa. Program je prenosiv ako se može manjim rutinskim poslom preraditi da funkcioniše na drugom računaru. Najpovoljniji slučaj bi bio potpuna nezavisnost programa od upotrebljenog računara. To se ostvaruje korišćenjem standardnih programskih jezika, ali praktično je teško ostvarljivo na kućnim računarima.

Program treba pisati struktuirano. Potrebno je pridržavati se pravila o disciplinovanom — struktuiranom programiranju. Time se ostvaruje da formiranje programa, tj. prvenstveno pisanje samog programa, bude što više rutinska aktivnost.

Standardni Komodorov bejzik, kao i mnogi drugi bejzik programski jezici ne podržavaju struktuirano programiranje, ali to svakako ne znači da ne treba disciplinovano pisati programe.

Dijagram toka

Formiranje preciznog algoritma, u cilju njegovog izražavanja programskih jezikom, za duže programe postaje nepregledno. Velika količina podataka i operacija nad podacima ne dozvoljava efikasan uvid, praćenje i menjanje. Radi bolje preglednosti koristi se grafičko predstavljanje algoritma.

Grafički simbolima se predstavljaju određene operacije, tj. radnje nad podacima. Unutar tih simbola se upisuju podaci i radnje nad podacima. Simboli su međusobno povezani linijama koje označavaju redosled obrade podataka. Takva struktura se naziva dijagram toka i ona prikazuje koje se operacije izvršavaju, i u kom redosledu. Dijagram toka se još naziva i algoritamska šema, a i programski blok dijagram.

Dijagram toka je grafički predstavljen algoritmom i kao takav je veoma pogodan za razvoj programa.

Prikazani su standardni grafički simboli koji se koriste za grafičko predstavljanje algoritma:

1. Početak i kraj programa se obeležavaju sledećim simbolom:



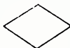
2. Unošenje podataka se predstavlja simbolom u obliku romba. Za sve operacije u kojima se podaci prenose iz spoljnog sveta u računar (upisivanje sa tastature, učitavanje sa kasetofona ili diska, rad sa palicom za igru,...) treba koristiti ovaj simbol:



3. Obrada podataka se predstavlja pravougaonikom. Pravougaonik se praktično koristi za sve operacije u kojima dolazi do transformacije tj. obrade podataka. To je najčešće korišćeni simbol u dijagramu toka.



4. Donošenje odluke se predstavlja rombom. Može se predstaviti i geometrijskim likom





prikazanim na slici. Koriste se za operacije koje dovode do grananja programa. To su logičke operacije čiji rezultat određuje dalji tok programa, tj. određuje kojim operacijama se nastavlja izvršavanje programa.

5. Izlaz podataka se predstavlja rombom. Koristi se u svim operacijama u kojima se podaci prenose iz računara u spoljni svet (ekran, zvučnik, disk,...).



Kao što je rečeno, unutar navedenih simbola upisuju se odgovarajući podaci i radnje. Pri tome se ne treba služiti programskim jezikom. Treba koristiti opis koji je nezavistan od programskog jezika koji će se koristiti.

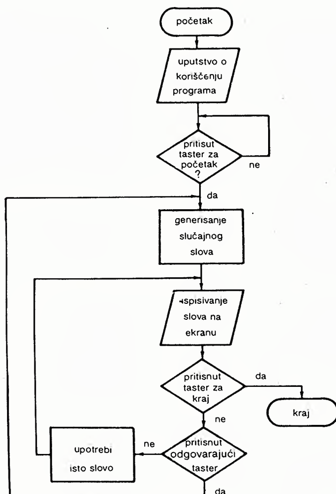
Poredak simbola koji označava tok izvršavanja treba biti od vrha naniže. Simboli se međusobno povezuju linijama koje označavaju tok izvršavanja. Pri tome treba nastojati da ne dođe do preseka linija.

Svaki korektan pristup razvijanju programa sadrži formiranje grafičkog predstavljanja algoritma — dijagrama toka. Vreme koje se utroši na formiranje dijagrama višestruko se vraća u fazi pisanja i testiranja programa. U slučaju dužih programa upotreba dijagrama toka je nezamenljiva.

Dat je dijagram toka za program koji pomaže učenju diktilografije.

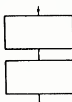
Osnovne programske strukture

Svaki program se može predstaviti pomoću tri osnovne programske strukture. To su:
- sekvenca
- selekcija
- interacija

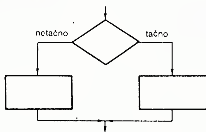


Sl. 5. 1. Dijagram toka programa za učenje daktilografije

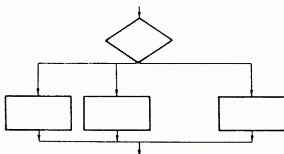
Sekvenca, ili linijska struktura, predstavlja niz naredbi koje se izvršavaju jedna za drugom. Odgovarajući grafički prikaz je sledeći:



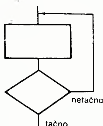
Selekcija, ili razgranata struktura, predstavlja mesto grananja u programu. Ostvaruje se naredbama **IF...THEN...ELSE** i **CASE**. U standardnom Komodorovom bejziku to se ostvaruje naredbama **IF...THEN** i **ON.** U Sajmons bejziku može se koristiti naredba **IF...THEN...ELSE.** Njen grafički prikaz je sledeći:



Za višestruko grananje u programu (CASE, ON) koriste se sledeći grafički simboli:



Iteracija ili petlja je struktura u kojoj se jedna ili više naredbi izvršavaju više puta za redom, sve dok ne bude zadovoljen uslov izlaska iz petlje. To se ostvaruje naredbama REPEAT...UNTIL i WHILE...DO. U standardnom Komodorovom bejziku to se može postići naredbama FOR...NEXT ili IF...THEN uz pomoć naredbe GOTO. U Sajmons bejziku moguće je koristiti naredbu REPEAT...UNTIL čiji je grafički prikaz dat.



Činjenica da se svaki algoritam može predstaviti gore navedenim strukturama je bitan element u razvoju programa. Poznavajući to, proces razvoja programa postaje bolje saglediv i lakše rešiv.

Testiranje programa

Nakon ispisivanja programa, program se unosi u računar. Zatim se pristupa njegovom testiranju. Testiranje ima za cilj da potvrdi funkcionalnost programa. Tokom testiranja potrebno je uočiti i otkloniti sve greške.

Testiranje se postupno sprovodi nad delovima programa koji predstavljaju logičke celine, a na kraju se sprovodi na kompletnom programu. Testiranje programa se obavlja njegovim izvršavanjem. Pri tome se mogu koristiti test podaci, a ne stvarni podaci. Takođe, ne mora se izvršavati ceo program već samo neki njegovi delovi.

U toku testiranja mogu se javiti greške nastale zbog nepravilnog korišćenja programskog jezika. To su takozvane greške sintakse, tj. greške pravila pisanja programa. Računar prijavljuje izveštajima o tome koja je greška u pitanju i na kom mestu u programu. Otklanjanje tih grešaka je relativno jednostavno. Svodi se na njihovo uočavanje i na upisivanje pravilno upotrebljenih naredbi programskog jezika.

Drugi tip grešaka koji se može javiti u programu su logičke greške (nazivaju se i greške semantike). One se javljaju kao rezultat pogrešne interpretacije problema. Mogu se javiti od samog početka sagledavanja problema, pa do upotrebe naredbi. Greška može biti u samom algoritmu, a takođe i u upotrebljenim naredbama iako je algoritam tačan. Otklanjanje ovih grešaka je svakako teži deo testiranja i razvoja programa.

Kada se isprave sve greške, program je kompletno testiran i spreman za izvođenje. Može se koristiti, kako je to uobičajeno na kućnim računarima, interpreterski, ili se može prevesti u mašinski program tj. može se kompilirati.

Dokumentacija

U pristupu programiranju od koga se želi efikasno i pouzdano dobijanje kvalitetnih programa vođenje dokumentacije je aktivnost na koju otpada do 50% kompletnog utrošenog vremena dobijanja programa. Navedeni procenat može izgledati veliki, ali u praksi potvrđuje svoju opravdanost. Uzimajući u obzir da je u procesu razvoja programa potrebno analizirati veliki broj činilaca bitnih za razvoj programa i da je potrebno doneti veliki broj odluka postaje jasno da se sve to ne može obaviti na apstraktnom nivou. Takođe, sam ispis programa — listing, pretstavlja zabeleženu samo poslednju fazu u dobijanju kompletnog programa.

Dokumentacija čini jasno i pregledno zapisivanje svih aktivnosti koje se obavljaju tokom dobijanja kompletnog proizvoda, u ovom slučaju programa. Potrebno ju je voditi od samog početka, do dobijanja programa.

Tokom razvoja programa potrebno je voditi radnu dokumentaciju čija je namena da omogući dalje aktivnosti i da zabeleži izvršene aktivnosti. Po dobijanju istestiranog programa potrebno je napisati završnu dokumentaciju. Njena namena je da omogući kasnije eventualne izmene i da omogući upotrebu delova razvijenog programa u budućim programima. Ona treba da do detalja objašnjava svaki deo programa i može se formirati na osnovu uredno vođenje radne dokumentacije. Završna dokumentacija treba da sadrži dijagrame toka, testove programa, podatke, načine testiranja, rezultate, poruke o greškama itd.

5.3 STRUKTUIRANO PROGRAMIRANJE

Struktuirano programiranje se može najkraće opisati ako se kaže da je to disciplinovano programiranje, programiranje prema dogovoru. Takav način programiranja je proizašao iz potrebe za poboljšavanjem postojećih načina programiranja.

Porast broja i složenosti problema koji su se postavljali pred programiranje uvelo je neophodnost za iznalaženjem metode programiranja koja će dati efikasniju — brzu izradu programa. Postavila se i potreba za preglednim i razumljivim programima. To je proizašlo iz potrebe za naknadnim modifikovanjem programa, u cilju njihovog poboljšavanja. Program koji nije napisan na način razumljiv drugim programerima, osim autoru programa, nije isplativ za modifikaciju.

Krajem šezdesetih i početkom sedamdesetih godina postavljena je osnova struktuiranog programiranja. Utvrđeni su zahtevi koje mora ispunjavati takvo programiranje. To su:

1. Korišćenje ograničenog broja precizno definisanih osnovnih programskih (kontrolnih) struktura.
2. Razvoj programa u koracima preciziranja.
3. Hijerarhijske ulazno — procesna — izlazna dokumentacija.
4. Modularnost programa.

Korišćenjem ograničenog broja precizno definisanih programskih struktura zavodi se red u programiranje. Smanjuje se broj mogućih rešenja nekog problema eliminisanjem proizvoljnih rešenja. Time se olakšava programiranje, a program postaje pregledniji samom autoru i ostalima. Pri tome treba imati na umu da programske strukture menjaju redosled izvršavanja operacija nad podacima, a ne rade obradu podataka.

Razvoj programa u koracima preciziranja

Struktuirano programiranje je programiranje u kome se razvoj programa odvija po nivoima — hijerarhiji. Pri tome se dobija i program u hijerarhijskom tj. struktuiranom obliku.

Kako je u prethodnoj tački izloženo, razvoj programa otpočinje apstraktnom analizom problema. Pažnja se usmerava na globalne aspekte problema, a za analizu se koristi govorni jezik.

Dalji razvoj programa se odvija od apstraktnog nivoa, u koracima preciziranja, ka raščlanjivanju apstraktnih podataka i radnji. Cilj raščlanjivanja je prevođenje algoritma u formu koja se može bez većih teškoća izraziti programskim jezikom. Svaki korak u tome čini jedan korak preciziranja. Posle određenog broja koraka algoritam je kompletno izražen programskim jezikom, tj. program je razvijen.

U svakom koraku preciziranja više kriterijuma određuje koje će se rešenje izabrati. Da li je to brzina izvršavanja, utrošena memorija, jasnoća programa ili nešto drugo, ostaje programeru na izboru. Moguće je da se na nekom nivou preciziranja ne može izabrati odgovarajuće rešenje, već da je to potrebno uraditi naknadno, nakon odlaska na dalje nivoe. Vraćanje koje se pri tome mora obaviti može odvesti do najviših nivoa apstrakcije, namećući izmene u samom polazu rešavanja.

Struktuirano programiranje se zbog načina programiranja naziva i programiranje od vrha na dole. Prednosti koje se dobijaju struktuiranim načinom su u dobijanju programa koji se lako menjaju. Te izmene mogu biti u cilju poboljšanja programa ili prilagođenja za drugi računar.

Procedure

Struktuiranim programiranjem, u koracima preciziranja, ostvaruje se da program bude razvijen, struktuiran u logičke celine. Deo programa koji predstavlja logičku celinu po funkciji koju obavlja, naziva se modulom. Klasični pristup programiranju takođe koristi podelu programa u module.

U struktuiranom programiranju uvodi se pojam procedure. Proceduru čini niz operacija određene namene. Osnovno svojstvo procedure je da se nizu operacija od kojih se sastoje može dodeliti ime. Dodeljivanje imena proceduri se naziva deklaracija procedure.

Procedura tj. niz operacija se može pozivati i izvršavati bilo gde u programu pozivom procedure preko njenog imena.

Može se uočiti sličnost između potprograma i procedure, ali razlika je suštinska. Dok je potprogram nastao sa ciljem uštede memorije, procedura je nastala iz sistematskog razvoja programa i nalazi se u programu i kada se samo jedanput koristi.

Svakako da je opravdano koristiti potprograme u smislu procedura u programima pisanim na programskim jezicima koji ne omogućavaju rad sa procedurama. Takav programski jezik je i standardni Komodorov bejzik. Za razliku od njega Sajmons bejzik omogućava rad sa procedurama kao i sa izvesnim brojem programskih struktura koje podržavaju struktuirano programiranje (videti poglavlje 6.2.6).

Razvoj programa otpočinje od apstraktnog nivoa. U toj fazi određeno je koje su najopštije funkcije koje program treba izvršavati. Uzmimo na primer da je potrebno napraviti program za crtanje. Na apstraktnom nivou je određeno da program treba izvršavati sledeće: crtanje, bojenje i brisanje. Svaka od ovih celina treba da predstavlja proceduru u programu. Pri tome logično je dodeliti ime proceduri koje odgovara njenoj nameni. Daljim razvojem programa svaka celina se precizira i dalje rasčlanjuje. Na primer crtanje se može sastojati iz: crtanje tačke, prave linije i zakrivljene linije. Ove nove celine su takođe procedure koje se pozivaju iz odgovarajućih prethodnih procedura.

Jedan poseban vid korišćenja procedure je kada procedura samu sebe poziva. To znači da je poziv procedure u samoj proceduri. Takva procedura se naziva rekurzivna procedura. Mnogi problemi se najefikasnije rešavaju upravo rekurzivnim metodama.

Lokalne i globalne promenljive

Procedura uvodi lokalne promenljive. To su promenljive koje su definisane prilikom deklaracije procedure. One postoje samo unutar procedure i za proceduru su lokalnog tipa. One su praktično bez ikakvog značaja van procedura. Svakako da procedura ne operiše samo sa lokalnim promenljivama, već može i sa promenljivama koje postoje u trenutku poziva procedure.

Nasuprot lokalnim promenljivama globalne promenljive su promenljive koje su definisane van procedure. Pojmovi lokalne i globalne su relativni tako da globalne promenljive neke procedure mogu biti lokalne za neku drugu proceduru. Treba uočiti da će pozivanje procedure imati za posledicu promenu vrednosti upravo globalnih promenljivih. Svakako da i lokalne promenljive menjaju vrednost, ali to nije od značaja pri pozivanju procedure.

Tokom izvršavanja procedure nije moguće pristupiti, tj. očitati ili promeniti vrednost globalnim promenljivama. Napuštanjem procedure globalne promenljive ponovo postaju dostupne.

Pojam lokalnog i globalnog se primenjuju ne samo na primenljive već i na druge objekte koji se javljaju u programu (funkcije, parametri, procedure).

Modularnost i adaptibilnost

Sistemskim pristupom programiranju, kao što je struktuirano programiranje, rešenja se dobijaju na jasan i sistematičan način. Dobija se program organizovan u logičke celine, module, sa jasnom funkcionalnošću programa kao celine i svakog modula posebno.

Time je dobijen program sa važnom osobinom da se može lako prilagoditi novim zahtevima. To se naziva adaptibilnost programa i pred današnjim zahtevima efikasnog razvoja programa je od primarnog značaja.

Izmene u programu u cilju prilagođenja mogu se obaviti promenom globalne strukture programa, uz zadržavanje istih modula. Eventualne izmene koje je potrebno izvršiti u samim modulima mogu se lakše izvršiti zbog jasne organizacije programa.

Ovim bi se završio prikaz principa programiranja. Umesto zaključka može se reći da je potrebno uložiti sve napore za ovladavanje struktuiranim načinom programiranja. Može se i reći da je jedna od mana programskog jezika bejzik ta što i površno napisani programi funkcionišu. Svakako da se na taj način ne može napredovati u ovladavanju programiranjem. Zbog toga je važno od samog početka bavljenja programiranjem pravilno usmeriti aktivnost, tj. ne navikavati se na pogrešan način rada. U radu sa Komodorom, Sajmons bejzik ima prednosti nad standardnim bejzikom, jer svojim programskim strukturama omogućuje bolju struktuiranost programa.

Veoma dinamična i zanimljiva oblast programiranja zahteva od onoga ko želi da se bavi programiranjem poznavanje postojećih znanja iz te oblasti, a takođe i praćenje novih. U tom pogledu, za dalji rad, navedena je adekvatna literatura na kraju knjige.

6
Sajmons bejzik

6.1 UVOD

Sajmons bejzik (engl. Simon's basic) je programski jezik koji je nastao kao proširenje standardnog bejzika sa ciljem da olakša iskoriščavanje velikih mogućnosti računara Komodor 64. To je ostvareno uvođenjem novih naredbi koje predstavljaju dopunu postojećih bejzik naredbi.

Sajmons bejzik donosi preko sto novih naredbi koje pokrivaju široki dijapazon primena. Moguće je izvršiti njihovo grupisanje po namenama za koje su nastale. Navedene su grupe i njima pripadajuće naredbe.

1. Pomoć u programiranju
- Olakšavanje upisivanja programa u računar (**AUTO, RENUMBER, MERGE**).
- Olakšavanje pregleda programa (**PAGE, OPTION, DELAY, FIND**).
- Upotreba funkcijskih tastera (**KEY, DISPLAY**).

2. Pomoć pri pojavi greške
- Nalaženje greške (**TRACE, RETRACE**).
- Pregled promenljivih (**DUMP**).
- Ponovno startovanje Sajmons bejzika (**COLD, OLD**).

3. Kontrola greške
- Preusmeravanje daljeg toka programa u slučaju pojave greške (**ON ERROR, NO ERROR, OUT**).

4. Zaštita programskih linija
- Onemogućenje prikazivanja programskih linija (**DISAPA, SECURE**).

5. Unošenje podataka
- Unošenje podataka preko tastature (**FETCH, INKEY**).
- Učitavanje podataka iz željene **DATA** linije (**RESET**).

6. Struktuirano programiranje
- Kontrolne strukture (**IF... THEN... ELSE, RCOMP, REPEAT... UNTIL, LOOP... EXIT IF... END LOOP**).
- Procedure (**PROC, END PROC, EXEC, CALL**).
- Promenljive (**LOKAL, GLOBAL**).
- Izračunati **GOTO** (**CGOTO**).

7. Rad sa brojevima

– Aritmetičke operacije (**MOD, DIV, FRAC, EXOR**).

– Rad sa binarnim i heksadecimalnim brojevima (**%, $**).

8. Rad sa stringovima

– Nove naredbe za rad sa stringovima (**INSERT, INST, PLACE, DUP**).

9. Ispisivanje rezultata

– Određivanje mesta ispisivanja i načina ispisivanja (**CENTRE, USE, AT, LIN, PAUSE**).

10. Boje na ekranu

– Treperenje ekrana (**FLASH, OFF, BFLASH, BFLASH O**).

– Postavljanje boja na ekranu (**COLOUR, BCKGNDS**).

– Popunjavanje ekrana tekstualnim znacima i inverzno ispisivanje (**FCHR, FCOL, FILL, MOVE, INV**).

11. Pomeranje sadržaja ekrana

– (**LEFT, RIGHT, UP, DOWN**).

12. Grafika

Za razumevanje naredbi za rad sa grafikom potrebna objašnjenja su izložena u naredbama: **HIRES, MULTI** i **LOW COL**. Detaljniji opis grafičkih načina rada Komodora je dat u poglavlju 9.

– Zadavanje grafičkog načina rada (**HIRES, MULTI, NRM, CSET, LOW COL, HI COL**).

– Crtanje i bojenje (**PLOT, TEST, LINE, REC, BLOCK, CIRCLE, ARC, ANGL, PAINT, DRAW, ROT**).

– Kombinovanje teksta sa grafikom (**CHAR, TEXT**).

13. Definisanje novih karaktera

– Stvaranje novih karaktera za tekstualni način rada (**MEM, DESIGN 2, @**).

14. Sprajtovi

Potrebna objašnjenja o sprajtovima su data u naredbi **DESIGN O, DESIGN 1.** Detaljniji uvid u rad Komodora sa sprajtovima dat je u poglavlju 9.

– Stvaranje sprajtova (**DESIGN O, DESIGN 1, @, MOB SET, MOB OFF, CMOB**).

– Upotreba sprajtova (**MMOB, RLOCMOB, DETECT, CHECK**).

15. Zvuk

Potrebna objašnjenja o upotrebi muzičkih sposobnosti Komodora su data u okviru izloženih naredbi ove grupe. Za detaljnija objašnjenja samog načina dobijanja zvuka mogu se pogledati poglavlja 9 i 10.

– Naredbe za dobijanje zvuka (**VOL, WAVE, ENVELOPE, MUSIC, PLAY**).

16. Rad sa diskom i kasetofonom

– Olakšavanje rada sa diskom (**DISK, DIR**).

– Snimanje sadržaja ekrana (**SCRSV, SCRLD**).

17. Rad sa štampačem

— Ispisivanje teksta sa ekrana i kopiranje ekrana (**HRDCPY, COPY**).

18. Rad sa upravljačkim uređajima.

— Rad sa svetlosnom olovkom i upravljačkom palicom (**PENX, PENY, POT, JOY**).

U narednom delu knjige su opisane naredbe i njihova primena po izloženim grupama i datim redosledom.

Velike mogućnosti Komodora postaju lako dostupne upotrebom Sajmons bejzika. Otpada potreba za velikim brojem **POKE** naredbi u standardnom bejziku kojima se programirao rad sa grafikom i zvuk. Posebno značajne su naredbe koje omogućavaju struktuiranje programa.

Pored svih prednosti Sajmons bejzik u raznim slučajevima neće omogućiti zadovoljavajuće rešenje. I pored velikog broja naredbi može se javiti potreba za nekim drugim naredbama. Takođe se može pojaviti i problem nedovoljne brzine izvršavanja programa. Posebno treba obratiti pažnju na korektnu upotrebu Sajmons bejzika jer u protivnom može doći do nepredvidivih pojava.

Sajmons bejzik se može nabaviti snimljen na traci, disketi ili u ROM modulu (kartridž). Po učitavanju u računar i startovanju, na ekranu se ispisuje poruka:

*****EXPANDED CBM V2 BASIC*****
30719 BASIC BYTES FREE

čime se javlja da je računar spreman za rad u Sajmons bejziku i da je za to na raspolaganju 30719 slobodnih bajtova memorije.

6. 2 NAREDBE I NJIHOVA UPOTREBA

6. 2. 1 Pomoć u programiranju

Olakšavanje upisivanja programa u računar. (AUTO, RENUMBER, MERGE)

AUTO

Namena: Automatsko dodeljivanje brojeva programskim linijama.

Opšti oblik: AUTO n, m

Argumenti:

n — broj prve linije u programu (od 0 do 63999).

m — korak između dve susedne linije, tj. broj koji se dodaje na broj prethodne linije da bi se dobio broj naredne linije (od 1 do 255).

Izvršenjem ove naredbe pritiskom na taster RETURN ispisuje se broj prve linije, a kursor se postavlja desno od njega. Upisivanjem naredbi i pritiskom na taster RETURN automatski se ispisuje broj druge linije i postupak unošenja programskih linija se može nastaviti. Završetak se ostvaruje unošenjem prazne linije tj. pritiskom tastera RETURN neposredno iza broja linije.

U slučaju broja programske linije većeg od 63999 prijaviće se izveštaj o grešci **?SYNTAX ERROR.**

Primer: AUTO 10,5

Prva linija u programu dobija broj 10, druga 15, treća 20, i tako redom sa korakom 5.

RENUMBER

Namene: Dodeljivanje novih brojeva programskim linijama.
Opšti oblik: **RENUMBER** n, m

Argumenti:

n – broj koji će dobiti prva linija u programu.
m – korak između dve susedne linije, tj. broj koji se dodaje na broj prethodne linije da bi se dobio broj naredne linije.

Napomena: Brojevi programskih linija u **GOTO** i **GOSUB** naredbama se ne menjaju!

Izvršenjem ove naredbe, po pravilu u direktnom načinu rada, svim programskim linijama dodeljuju se novi brojevi. Prva linija u programu dobija broj n, a svaka naredna za korak m veći. Time se ostvaruje mogućnost umetanja novih linija u program koji se razvija, a takođe se poboljšava njegova preglednost.

Brojevi programskih linija dobijeni na ovaj način mogu biti veći od regularnih 63999, ali ne i veći od 65535. Njih treba izbegavati jer su nedostupni standardnom Komodorovom bejziku.

Upotrebom simboličkih adresa i procedura u Sajmons bejziku (videti: Struktuirano programiranje) eliminiše se nepogodnost da naredba **RENUMBER** ne menja adrese u **GOTO** i **GOSUB** naredbama.

Primer:

```
1 FOR N=1 TO 127
2 :PRINT CHR$(N);
7 NEXT N
```

Nakon unošenja date tri linije i izvršenjem naredbe **RENUMBER 100,10** prva linija će dobiti broj 100, druga 110 i treća 120.

MERGE

Namena: Učitavanje programa bez brisanja postojećeg.
Opšti oblik: MERGE „ime", P

Argumenti:

ime – ime programa koji se učitava.
P – broj perifernog uređaja sa koga se učitava program (za disk jedinicu je 8, a za kasetofon 1 i ne mora se navesti).

Program koji se učitava nadovezuje se na postojeći program u memoriji računara. Tako se može desiti da programske linije sa većim brojem prethode linijama sa manjim brojem. Tada je potrebno upotrebiti naredbu za dodeljivanje novih brojeva programskim linijama **(RENUMBER).**

Olakšavanje pregleda programa. (PAGE, OPTION, DELAY, FIND)

PAGE

Namena: Deljenje programa na stranice (engl. page) za vreme njegovog prikazivanja (listanja) na ekranu.
Opšti oblik: PAGE n

Argumenti: n – broj za jedan manji od broja redova koji se prikazuju na ekranu. Može biti od 0 do 255, ali za praktičnu primenu koristi se od 0 do 23.

Izvršenjem ove naredbe, uobičajeno u direktnom načinu rada, određuje se broj redova ekrana za prikazivanje programa. Izvršenjem naredbe **LIST** prikazivaće se program u delovima čija je dužina određena naznačenim brojem redova ekrana. Prelazak sa jedne na drugu grupu ostvaruje se pritiskom tastera **RETURN.**

Ukoliko se želi prekinuti prikazivanje programa, to se ostvaruje pritiskom na taster **RUN STOP** u toku ispisivanja programa na ekranu. Privremeno zaustavljanje ispisivanja ostvaruje se pritiskom tastera **C**=. Povratak u normalan način prikazivanja programa ostvaruje se naredbom **PAGE 0.**

Primer: PAGE 5

Program će se prikazivati u grupama od po 6 redova ekrana.

OPTION

Namena: Prikazivanje svih naredbi Sajmons bejzika inverzno.
Opšti oblik: OPTION n

Argumenti: n – broj koji određuje način prikazivanja naredbi Sajmons bejzika. Ako je n = 10 naredbe će biti prikazane inverzno, a ako je n<>10 biće prikazane normalno.

Primer: OPTION 10

Naredbe se prikazuju inverzno.

Primer: OPTION 00

Naredbe se prikazuju normalno

DELAY

Namena: Promene prikazivanja programa (listanja).
Opšti oblik: DELAY n

Argumenti: n – broj koji određuje brzinu ispisivanja teksta programa na ekranu (od o do 255).

Naredba se po pravilu zadaje direktno. U sledećoj tabeli su date brzine ispisivanja karaktera na ekranu za neke vrednosti argumenta.

n	karaktera/sekundi
1	290
5	100
10	50
20	25

Dozvoljena je i vrednost argumenta 0 koja daje najmanju brzinu. Praktično odgovara vrednosti argumenta 256.

Primer: DELAY 40

FIND

Namena: Nalaženje (engl. find) programskih linija u kojima je naznačeni kôd ili niz kodova, ili nalaženje programskih linija u kojima je naznačeni string.

Opšti oblik: **FIND** kôd
FIND „string"

Argumenti: kod — kôd ili niz bejzik kodova (slova, brojevi, znaci, imena naredbi).
string — karakter ili niz karaktera koji mogu činiti string. Zadaju se unutar navodnika.

Napomene: Ako se između reči **FIND** i navedenog koda, tj. stringa nalazi prazno polje, ono će biti uključeno u traženje niza.

Prazna polja u programu, iza brojeva programskih linija, ne uzimaju se u obzir prilikom pretraživanja.

Upotrebom ove naredbe biće ispisani brojevi svih linija u kojima se nalazi niz kodova ili string naveden u naredbi **FIND**. Naredbe koje se traže mogu biti zadate u skraćenom obliku. Za nalaženje stringa ili dela stringa potrebno je otvoriti navodnike i navesti string. Zatvaranje navodnika nije obavezno, a takođe i ispisivanje celog stringa, sem u slučaju da se traži strogo određeni string.

Primer:

```
10 PRINT CHR$(147)
20 PRINT "KOLIKO JE 7*8"
30 INPUT A
40 IF A=56 THEN PRINT"TACNO":END
50 PRINT "POGRESNO":GOTO 10
```

Nakon unošenja gornjeg programa sledeće naredbe će dati navedene rezultate:

naredbe	brojevi programskih linija			
FIND?	10	20	40	50
FINDCHR				
FIND147	10			
FIND „KOL	20			
FINDA	30	40		
FIND „TACNO"	40			
FIND „POGR	50			

Naredbom **FINDCHR** nije nađena reč **CHRS** u liniji 10, koja je i u računaru zapamćena kao jedan kôd.

Upotreba funkcijskih tastera (KEY, DISPLAY)

KEY

Namena: Dodeljivanje niza karaktera ili naredbi funkcijskim tasterima.

Opšti oblik: **KEY** n, „string"

Argumenti: n — broj funkcijskog tastera (od 1 do 16).
string — string koji se dodeljuje tasteru n (do 15 karaktera tj. kodova).

Napomene: Kontrolni karakteri se unose u string pomoću naredbe **CHRS** (kôd karaktera).

Dodeljivanje praznog stringa funkcijskom tasteru neće obrisati prethodno dodeljeni string, već će dodeliti 15 nedefinisanih karaktera.

Ova naredba omogućuje da se funkcijskim tasterima (četiri krajnja desna tastera) dodele posebne namene. Moguće im je dodeliti niz karaktera koji se često pojavljuju u pro-

gramu, koje nakon toga nije potrebno unositi karakter po karakter, već je dovoljno pritisnuti odgovarajući funkcijski taster. Takođe im se mogu dodeliti i jedna ili više bejzik naredbi.

Na raspolaganju je 16 funkcijskih tastera čije se pozivanje ostvaruje pritiskom sledećih tastera:

funkcijski taster	upotrebiti tastere	funkcijski taster	upotrebiti tastere
f1	f1	f9	C= f1
f2	SHIFT i f1	f10	SHIFT C= f1
f3	f3	f11	C= f3
f4	SHIFT i f3	f12	SHIFT C= f3
f5	f5	f13	C= f5
f6	SHIFT i f5	f14	SHIFT C= f5
f7	f7	f15	C= f7
f8	SHIFT i f7	f16	SHIFT C= f7

Primeri: **KEY 1, "PRINT"**
KEY 2, CHR$ (147) + "LIST" + CHR$ (13)

U prvom primeru ostvareno je da svaki pritisak na taster f1 ispisuje **PRINT.** U drugom primeru ostvareno je da se svakim pritiskom tastera f2 postiže isti efekat kao i zadavanje i izvršenje naredbi brisanja ekrana **(SHIFT/CLR HOME)** i **LIST.**

DISPLAY

Namena: Prikaz stingova dodeljenih funkcijskim tasterima.

Opšti oblik: **DISPLAY**

Primer: **DISPLAY**

Izvršenjem ove naredbe nakon izvršenja primera za naredbu **KEY** dobiće se:

KEY1, "PRINT"
KEY2, "♥LIST" + CHR$(13)
KEY3,""
KEY4,""
:
:
KEY15,""
KEY16,""

6.2.2 **Pomoć pri pojavi greške**

Nalaženje greške, (TRACE, RETRACE)

TRACE

Namena: Ispitivanje brojeva programskih linija koje se izvršavaju.

Opšti oblik: **TRACE n**

Argumenti: n broj koji određuje da li će se ispisivati brojevi programskih linija. Ako je n 10 ispisivaće se, a ako je n<>10 neće.

Napomene: Ako računar radi u grafici visoke rezolucije, ili je u višebojnom načinu rada (HRG MC), neće se prikazivati brojevi programskih linija.

Ovo je vrlo korisna naredba jer omogućava praćenje izvršavanja programa. U gornjem desnom uglu ekrana, u polju 6 puta 6 karaktera, ispisuju se brojevi programskih linija koje se izvršavaju nakon startovanja programa. Time je omogućeno praćenje toka programa, što može biti od koristi naročito u dužim programima.

S obzirom da Sajmons bejzik ne raspolaže naredbom koja bi zaustavljala program nakon svake izvršene naredbe, brzina ispisivanja brojeva linija je najčešće velika da bi se pogodno mogao pratiti tok izvršavanja programa. U tom slučaju je potrebno usporiti izvršavanje programa uvođenjem pauza, tj. petlji za usporenje.

Primer: **TRACE 10**

Uključivanje funkcije **TRACE.**

RETRACE

Namena: Ispisivanje poslednjeg sadržaja polja u kome se ispisuju brojevi programskih linija dobijeni upotrebom naredbe **TRACE.**

Opšti oblik: **RETRACE**

Ovom naredbom se omogućuje dobijanje poslednjih šest brojeva programskih linija koje su izvršene pre zaustavljanja programa, što može biti od koristi ako su iz bilo kog razloga prethodno izgubljene.

Primer: **RETRACE**

Pregled promenljivih, (DUMP)

DUMP

Namena: Ispisivanje promenljivih koje se koriste u programu i njihovih trenutnih vrednosti.

Opšti oblik: **DUMP**

Napomene: Višedimenzionalne promenljive, brojne ili string (npr. A(1) i A$(5)), se ne prikazuju.

Negativne vrednosti realnih promenljivih prikazuju se pozitivno.

Negativne vrednosti celobrojnih promenljivih (npr. A%) prikazuju se uvećane za 65536 (2↑16).

Prazan string (npr. A$="") prikazuje se kao niz od 256 nedefinisanih znakova.

Ispisivanje promenljivih i njihovih vrednosti može se usporiti pritiskom na taster **CTRL.**

Primer: **A=5:B=−6:A%=7:B%=−8:A$=„MIKRO":B$=""**

Izvršiti naredbu **DUMP** nakon dodele vrednosti promenljivama.

Ponovno startovanje Sajmons bejzika (COLD, OLD)

COLD

Namena: Hladan start Sajmons bejzika.

Opšti oblik: **COLD**

Izvršenjem ove naredbe računar se prevodi u stanje kao da je Sajmons bejzik ponovo učitan. Ispisuje se poruka: *** **EXPANDED CBM V2 BASIC *** 30719 BASIC BYTES FREE**, a sistemske promenljive, vektori i pokazivači se postavljaju na početne vrednosti. Bejzik program se ne briše i moguće ga je povratiti naredbom **OLD**, pod uslovom da u međuvremenu nije uneta ni jedna programska linija.

Primer: Videti primer za naredbu **OLD.**

OLD

Namena: Vraćanje programa izbrisanog naredbama **COLD** ili **NEW**
Opšti oblik: OLD

Izvršenjem naredbi **COLD** ili **NEW** ostvaruje se brisanje bejzik programa. Pri tome on nije fizički obrisan iz RAM memorije što omogućava njegovo ponovno dobijanje naredbom **OLD**. Uslov je da posle naredbi **COLD** ili **NEW** nije upisana nova programska linija.

Primer:

```
10 REM PRIMER ZA
20 REM COLD I OLD
```

Upisivanjem datog programa i njegovim brisanjem naredbama **COLD** ili **NEW** program se može povratiti izvršenjem naredbe **OLD.** To se, zatim, može proveriti zadavanjem i izvršenjem naredbe **LIST.**

6.2.3 Kontrola greške

Preusmeravanje daljeg toka programa u slučaju pojave greške (ON ERROR, NO ERROR, OUT)

ON ERROR

Namena: Nastavljanje programa, u slučaju pojave greške, od zadate programske linije.

Opšti oblik: ON ERROR: GOTO n

Argumenti: n — broj programske linije od koje se nastavlja program u slučaju pojave greške.

Napomene: Naredbu treba koristiti samo programski.

Ova naredba omogućava da se ne prekine izvršavanje programa u slučaju pojave greške. Program će se nastaviti od navedene programske linije. Od te linije pa nadalje uobičajeno se smešta deo programa koji je zadužen za analizu nastale greške i donošenje odluke o daljem toku izvršavanja programa. To je omogućeno time što se u programske promenljive rezervisane od strane Sajmons bejzika smešta broj greške koja se pojavila (promenljiva ERRN) i broj programske linije u kojoj se greška pojavila (promenljiva ERRLN).

Greške koje se mogu javiti i njima odgovarajući brojevi grešaka su:

broj greške	greška
1	TOO MANY FILES
2	FILE OPEN
3	FILE NOT OPEN
4	FILE NOT FOUND
5	DEVICE NOT PRESENT

10	NEXT WITHOUT FOR
11	SYNTAX
12	RETURN WITHOUT GOSUB
13	OUT OF DATA
14	ILLEGAL QUANTITY
15	OVERFLOW
16	OUT OF MEMORY
17	UNDEFINED STATEMENT
18	BAD SUBSCRIPT
19	RE – DIMENSIONED ARRAY
20	DIVISION BY ZERO
21	ILLEGAL DIRECT
22	TYPE MASMATCH
23	STRING TOO LONG

Iz datog spiska vidi se da nisu navedene sve greške, što znači da nije moguće primeniti ovu naredbu za kontrolu svake greške.

Prekidanje dejstva ove naredbe ostvaruje se naredbom **NO ERROR.**

Primer:

```
10 PRINT CHR$(147)
20 ON ERROR:GOTO 300
30 X=10
40 REPEAT
50 : PRINT 10"/"X"="10/X
60 : X=X-1
70 UNTIL X=-11
80 NO ERROR
90 END
300 IF ERRN=20 THEN PRINT"NEDEFINISANA VREDNOST":X=X-1:GOTO 40
310 NO ERROR
320 END
```

U toku izvršavanja datog programa javlja se greška usled nedozvoljenog deljenja sa nulom. Njenom pojavom program se nastavlja linijom 300, u kojoj se ispisuje predviđena poruka, a takođe se obezbeđuju dalji uslovi za izvršavanje programa.

U slučaju da se očekuje više različitih grešaka pogodno je upotrebiti naredbu **CGOTO** za odlaske na različite programske linije u zavisnosti od broja greške (npr. **CGOTO 300 + 10*ERRN).** Istotako se može upotrebiti i promenljiva ERRLN.

NO ERROR

Namena: Prekid dejstva naredbe **ON ERROR.**

Opšti oblik: **NO ERROR**

Ovom naredbom zaustavlja se dalje dejstvo prethodno izvršene naredbe **ON ERROR.** Izvršavanje programa se po pojavi greške ponovo zaustavlja i ispisuje se odgovarajući izveštaj o grešci.

Ovu naredbu treba koristiti ako je prethodno zadata naredba **ON ERROR.** U slučaju da je program zaustavljen (npr. **RUN STOP**), a da se prethodno nije stigla izvršiti naredba **NO ERROR** treba je zadati direktno.

Primer: Videti primer za naredbu **ON ERROR.**

OUT

Namena: Ispisivanje izveštaja o grešci poslednje greške.

Opšti oblik: OUT

Napomene: Neće se ispisati broj linije u kojoj se pojavila greška, već broj linije u kojoj je navedena **OUT** naredba.

Izvršenjem ove naredbe prekida se dalje izvršavanje programa.

Primer: U primeru za naredbu **ON ERROR** dodati liniju:
85 OUT

6.2.4 Zaštita programskih linija

Onemogućenje prikazivanja programskih linija (DISAPA, SECURE)

DISAPA

Namena: Obeležavanje programskih linija koje se žele učiniti nevidljivim.

Opšti oblik: DISAPA

Uvođenje ove naredbe iza broja programske linije učiniće da cela linija, sem njenog broja, bude, po izvršenju naredbe **SECURE 0,** nevidljiva. Unošenjem naredbe, računar će iza nje automatski ubaciti četiri dvotačke (:).

Primer:

```
10 INPUT "UNESITE KONTROLNI KOD";A$
20 IF A$<>"THX-99"THEN PRINT "POGRESNO":END
30 PRINT "TACNO"
```

U datom programu želi se zaštititi linija 20. U tom cilju ubacuje se naredba za obeležavanje:

```
20 DISAPA:IF A$-"THX-99" THEN PRINT "POGRESNO":END
```

Program nakon toga izgleda:

```
10 INPUT "UNESITE KONTROLNI KOD";A$
20 DISAPA::::IF A$<>"THX-99" THEN PRINT "POGRESNO":END
30 PRINT "TACNO
```

SECURE

Namena: Zaštita programskih linija obeleženih naredbom **DISAPA.**

Opšti oblik: SECURE 0

Izvršenjem naredbe, obeležene programske linije postaju, sem brojeva linija, nevidljive.

Primer: Program dat u opisu naredbe **DISAPA** nakon izvršenja naredbe **SECURE 0** postaje:

10 INPUT "UNESITE KONTROLNI KOD"; A$
20
30 PRINT "TACNO"

6.2.5 **Unošenje podataka**

Unošenje podataka preko tastature (FETCH, INKEY)

FETCH

Namena: Učitavanje sa tastature samo željenih karaktera.
Opšti oblik: FETCH „string", J, v

Argumenti: string – string kojim se zadaju karakteri koji će se učitavati.
J – broj karaktera koji će se učitati.
v – promenljiva kojoj se dodeljuju učitani karakteri.

Ovom naredbom se sa tastature učitavaju, ne svi, već samo naznačeni karakteri. U toku izvršavanja programa nailaskom na ovu naredbu, slično naredbi **INPUT**, ne prelazi se na dalje izvršavanje naredbi sve dok se ne pritisnu određeni tasteri i na kraju pritisne taster **RETURN.** Za razliku od naredbe **INPUT**, da bi se program nastavio, potrebno je uneti prethodno naznačene karaktere, i to onoliko koliko ih je naznačeno argumentom J.

Promenljiva može biti string ili brojna. U prvom slučaju se mogu prihvatati svi tasteri, a u drugom samo tasteri sa ciframa. Preporučljivo je koristiti samo string promenljivu.

String kojim se zadaju karakteri može biti dat i u obliku promenljive ili izraza. Navođenjem sledećih vrednosti prihvataće se sa tastature sledeće:

CHR$(17) (pokazivač dole) – karakteri čiji su kodovi od 32 do 64 (cifre i posebni znaci).
CHR$(19) (CLR/HOME) – karakteri čiji su kodovi od 65 do 90 (velika slova).
CHR$(29) (pokazivač desno) – velika slova sa i bez pritisnutog tastera **SHIFT.**

Primer:
```
10 PRINT CHR$(147)
20 PRINT "OOGOVORITE SA OA,NE ILI MOZOA"
30 FETCH"OANE",2,A$
40 IF A$-"OA" THEN PRINT "O.K. IOEMO OALJE":STOP
50 IF A$-"NE" THEN PRINT "ONDA NE IOEMO":STOP
60 GOTO 20
```

INKEY

Namena: Određivanje koji funkcijski taster je pritisnut.
Opšti oblik: INKEY

Ovo je funkcijska naredba. Ona daje brojnu vrednost (od 1 do 16) koja odgovara broju pritisnutog funkcijskog tastera (videti objašnjenja za naredbu **KEY**). Ako nije pritisnut ni jedan taster, dobija se vrednost nula.

Primer:
```
10 PRINT CHR$(147)
20 A-INKEY:IF A-O GOTO 10
30 PRINT "PRITISNUT JE TASTER F";A
40 GOTO 10
```

Po startovanju programa, pritiskom na neki od funkcijskih tastera ispisaće se koji je pritisnut.

Učitavanje podataka iz željene DATA linije (RESET)

RESET

Namena: Očitavanje vrednosti iz **DATA** naredbe od željene linije.

Opšti oblik: RESET n

Argumenti: n — broj programske linije od koje će se čitati podaci iz **DATA** naredbe.

U standardnom Komodorovom bejziku naredbom **READ** čitaju se podaci iz datoteke, ali pri tome se počinje od **DATA** linije sa najmanjim brojem. Naredbom **RESET** očitavanje počinje od naznačene programske linije.

Primer:

```
10 PRINT CHR$(147)
20 PRINT "PRITISNITE"
30 PRINT " 1 ZA BROJEVE"
40 PRINT " 2 ZA SLOVA"
50 FETCH "12",1,A$
60 IF A$="1" THEN RESET 200:GOTO 80
70 RESET 210
80 I=0
90 REPEAT
100 : I=I+1
110 : READ A$(I):PRINT A$(I)
120 UNTIL A$(I)="*"
130 END
200 DATA 1,2,3,4,5,6,7,*
210 DATA A,B,C,D,E,F,G,H,I,*
```

U datom primeru biraju se i očitavaju podaci iz **DATA** naredbe (datoteke) u liniji 200 ili 210.

6.2.6 Struktuirano programiranje

Kontrolne strukture (IF... THEN... ELSE, RCOMP, REPEAT... UNTIL, LOOP... EXIT, IF... END LOOP)

IF....THEN...ELSE

Namena: Grananje u programu u zavisnosti od rezultata izraza. Ako je izraz tačan, izvršava se jedna radnja, a ako je netačan, druga.

Opšti oblik: IF b **THEN** s1 **ELSE** s2

Argumenti: b — logički izraz. Njegovim izračunavanjem dobija se logički rezultat koji može biti istinit ili lažan.

s1 — radnja, tj. naredba ili grupa naredbi koja se izvršava ako je rezultat izraza tačan (istinit).

s2 — radnja, tj. naredba ili grupa naredbi koja se izvršava ako je rezultat izraza netačan (lažan).

Napomene: Ispred i iza reči **ELSE** potrebno je staviti dvotačke (:).

Ako izvršavanje radnje s2 nije potrebno, tj. ako ona ne postoji, može se izostaviti reč **ELSE** čime naredba prelazi u standardnu Komodorovu bejzik naredbu **IF** b **THEN** s1.

Ova naredba je proširena standardna naredba **IF THEN**, na taj način da omogućuje struktuiranje bejzik programa. Služi za formiranje jedne od osnovnih programskih struktura – selekcije.

Primer:

```
10 PRINT CHR$(147)
20 PRINT "KOLIKO JE 12 PUTA 12";
30 INPUT A
40 IF A=144 THEN PRINT"TACNO":END:ELSE:PRINT "NIJE TACNO"
45 PAUSE 1:GOTO 10
50 END
```

U zavisnosti od datog odgovora ispisuju se potrebni izveštaji (TAČNO ili NIJE TAČNO). Treba uočiti da se davanjem tačnog odgovora izvršavanje primera zaustavlja naredbom **END** u liniji 40, što praktično znači da se naredba **END** u liniji 50 neće nikada izvršiti.

RCOMP

Namena: Ispisivanje u novim programskim linijama naredbi koje se izvršavaju u zavisnosti od rezultata izraza prethodno izvršene **IF THEN ELSE** naredbe.

Opšti oblik: RCOMP:s1:**ELSE**:s2

Argumenti: s1 — grupa naredbi koja se izvršava ako je rezultat izraza prethodno izvršene naredbe tačan (istinit).

s2 — grupa naredbi koja se izvršava ako je rezultat izraza prethodno izvršene naredbe netačan (lažan).

Napomene: Iza reči **RCOMP** potrebno je staviti dvotačku (:). Takođe ispred i iza reči **ELSE**.

Ako izvršavanje naredbi s2 nije potrebno, tj. ako one ne postoje, može se izostaviti reč **ELSE**, čime naredba prelazi u oblik

RCOMP:s1

Ako nije potrebno izvršavanje naredbi s1, ova naredba dobija oblik

RCOMP:ELSE:s2

Ova naredba oslobađa od ograničenja Komodorovog ekranskog editora. Omogućuje ispisivanje jednog niza naredbi u više programskih linija.

Primer:

```
10 PRINT CHR$(147)
20 PRINT "ZELITE LI DA VAM NESTO POKAZEM","DA/NE";
30 INPUT A$
40 IF A$="DA" THEN FOR I=0 TO 15:FOR J=0 TO 15:ELSE:PRINT "IAKO";
50 RCOMP:COLOUR I,J:ELSE:PAUSE 1
60 RCOMP:NEXT:NEXT:ELSE:PRINT" NE ZELITE, NESTO STE NAUCILI"
70 RCOMP:PRINT "HVALA"
```

REPEAT...UNTIL

Namena: Formiranje jedne od osnovnih programskih struktura: petlje (iteracije) sa izlaskom na dnu.

Opšti oblik: REPEAT:s:**UNTIL** b

Argumenti: s — grupa naredbi koja se izvršava sve dok nije zadovoljen izraz b.

b — logički izraz čiji rezultat određuje izvršavanje grupe naredbi. Ako je njegov rezultat netačan (lažan), izvršavaće se grupa naredbi s. Ako je tačan (istinit), preći će se na dalje izvršavanje programa.

Ovom naredbom se formira petlja kao i standardnom **FOR NEXT** naredbom, ali od nje pruža više mogućnosti jer formira uslovnu petlju. Petlja, i naredbe u njoj, neće se izvršavati zadati broj puta, već sve dok se ne zadovolji uslov b.

Za petlju dobijenu ovom naredbom karakteristično je da se prvo izvršava grupa naredbi s, a zatim se proverava uslov b za dalje izvršavanje, odnosno napuštanje petlje. Otuda naziv petlja sa izlaskom na dnu.

Primer:

```
10 A=1
20 REPEAT
30 : S=S+A
40 : A=A+1
50 UNTIL A>100
60 PRINT S
```

Iz datog primera u kome se nalazi suma prvih 100 celih pozitivnih brojeva vidi se da početak petlje određuje reč **REPEAT**, a njen kraj reč **UNTIL**. Grupu naredbi u petlji čine naredbe u linijama 30 i 40. Uslov za napuštanje petlje je da je A veće od 100.

LOOP...EXIT IF...END LOOP

Namena: Formiranje petlje sa izlaskom u sredini.
Opšti oblik: LOOP: s1 **:EXIT IF** b: s2 **:END LOOP**

Argumenti: s1 — grupa naredbi koja se izvršava sve dok nije zadovoljen izraz b.
b — logički izraz čiji rezultat određuje dalje izvršavanje petlje. Ako je njegov rezultat netačan (lažan), naredbe s1 i s2 će se i dalje izvršavati. Ako je njegov rezultat tačan (istinit), napustiće se dalje izvršavanje petlje.
s2 — grupa naredbi koja se izvršava sve dok nije zadovoljen izraz b.

Napomene: Između početka petlje koji je označen sa **LOOP**, i kraja petlje označenog sa **END LOOP**, može se nalaziti više izlaza iz petlje, označenih sa **EXIT IF**. Između tih izlaza mogu se nalaziti naredbe.

Naredbe s1 ili s2 mogu se izostaviti čime se dobija petlja sa izlaskom na dnu ili petlja sa izlaskom na vrhu.

Primer:

```
10 LOOP
20 : EXIT IF A>100
30 : Z=Z+A
40 : A=A+1
50 END LOOP
60 PRINT Z
```

U datom primeru izračunava se zbir prvih 100 celih pozitivnih brojeva. Početak petlje je u liniji 10, izlaz iz petlje je u liniji 20, i kraj petlje u liniji 50. Upoređujući primer sa opštim oblikom ove naredbe zaključuje se da u primeru ne postoji grupa naredbi s1. Time je dobivena petlja sa izlaskom na vrhu. Ubacivanjem linije:

15 PRINT AT (30,5) Z

dobija se petlja sa izlaskom u sredini.

Procedure (PROC, END PROC, EXEC, CALL)

PROC

Namena: Dodeljivanje imena proceduri.
Opšti oblik: PROC ime
Argumenti: ime — ime koje se dodeljuje proceduri.

Napomene: Sve što je navedeno do kraja programske linije iza reči **PROC** biće ime procedure.

Koncept procedure je da se program podeli u logičke celine, blokove, tj. module koji će omogućiti strukturiranje programa. Početak procedure je određen ovom naredbom, a njen kraj naredbom **END PROC.** Izvršavanje procedure se obavlja pozivanjem njenog imena pomoću, predviđenih naredbi **EXEC** i **CALL.**

Primer:

```
100 PROC UNOS PODATAKA
110 : INPUT "PREZIME I IME";PI$
120 : INPUT "ULICA I BROJ";UB$
130 : INPUT "MESTO";MS$
140 : INPUT "DIN.";DN
150 END PROC
```

Dati primer predstavlja deo nekog većeg programa. Ova celina je procedura za unošenje izvesnih podataka o korisnicima. Njeno ime je UNOS PODATAKA.

END PROC

Namena: Označavanje kraja procedure.
Opšti oblik: END PROC

Napomene: U slučaju izvršavanja ove naredbe bez prethodnog pozivanja procedure pomoću naredbe **EXEC,** prijaviće se izveštaj o grešci **END PROC WITHOUT EXEC.** Ova naredba odgovara naredbi **RETURN** standardnog bejzika.

Primer: Videti primer za **PROC.**

EXEC

Namena: Izvršavanje procedure.
Opšti oblik: EXEC ime

Argumenti: ime — ime procedure koju treba izvršiti.

Napomene: Razlikovati od naredbe **CALL**

Izvršavanjem ove naredbe program se nastavlja naznačenom procedurom, čiji se početak označava sa **PROC** ime. Nakon izvršenja procedure i poslednje naredbe u njoj, **END PROC,** program se nastavlja od naredbe koja sledi ovde izloženu naredbu **EXEC,** kojom je procedura pozvana. Naredba odgovara naredbi **GOSUB** standardnog bejzika.

Ime procedure predstavlja njenu simboličku adresu, koja se naziva labelom.

Primer:

```
500 PROC SORT IMENA
510 REPEAT
520 : P=O
530 : FOR N=1 TO N-1
540 :  IF I$(N)<=I$(N+1) THEN CALL NOVO IME
550 :  Z$=I$(N):I$(N)=I$(N+1):I$(N+1)=Z$:P=1
560 :  PROC NOVO IME
570 : NEXT
580 UNTIL P=O
590 ENO PROC
```

Procedura data u primeru obavlja sortiranje imena ili bilo kakvih stringova. Sortiranje je postupak dovođenja stringova u neki red. U ovom slučaju radi se o sortiranju po abecedi. Primenjen je takozvani šel (eng. shell) način sortiranja. Upoređuju se dva susedna imena i ako je potrebno međusobno se dovode u red, tj. zamenjuju im se mesta. Postupak se obavlja od prvog do poslednjeg imena, sve dok je to potrebno.

Dati primer predstavlja deo nekog većeg programa. Da bi mogao funkcionisati, prethodno je potrebno uneti imena u višedimenzionalnu promenljivu (niz) I$. Izvršava se pozivom imena naredbom:

EXEC SORT IMENA

koja može biti smeštena bilo gde u glavnom programu. Poređenje imena obavlja se u liniji 540, a ako je potrebna njihova zamena, to se obavlja u liniji 550. Parametar P će se postavljati na nulu sve dok u jednom prolazu kroz sva imena bude barem jedne zamene, što će označavati potrebu za novim prolazom.

CALL

Namena: Izvršavanje procedure.
Opšti oblik: CALL ime

Argumenti: ime – ime procedure od koje se nastavlja dalje izvršenje programa.

Napomene: Razlikovati od naredbe **EXEC.**
Izvršavanjem ove naredbe program se nastavlja naznačenom procedurom, čiji se početak označava sa **PROC** ime. Nakon izvršenja procedure i poslednje naredbe u njoj, **END PROC,** program se nastavlja od sledeće naredbe. Ova naredba odgovara naredbi **GO TO** standardnog bejzika.

Primer: Videti primer za naredbu **EXEC.**

Promenljive (LOCAL, GLOBAL)

LOCAL

Namena: Deklaracija lokalnih promenljivih.
Opšti oblik: LOCAL a, b, c...

Argumenti: a, b, c... – promenljive koje se uvode kao lokalne.

Lokalne promenljive koriste se samo u okviru procedura i pružaju veću slobodu u programiranju za razliku od standardnog bejzika u kome postoje samo globalne promenljive. Lokalne promenljive okončavaju svoje postojanje po izvršenju za njih predviđenih zadataka. To program čini preglednijim, omogućuje njegovo struktuiranje i podelu u module, a takođe dovodi do uštede u memoriji.

Primer:
```
10 A=2.71:A$="ABC":B%=3:C=-13
20 LOCAL A,A$,B%
30 A=5.1:A$="QWERTY":B%=1000
40 PRINT A,A$,B%,C
50 GLOBAL
60 PRINT A,A$,B%,C
```

Dati primer nema veću upotrebnu vrednost, ali prikazuje kako promenljive sa istim imenom mogu imati dve vrednosti, lokalnu i globalnu.

GLOBAL

Namena: Vraćanje globalnih vrednosti promenljivama.
Opšti oblik: GLOBAL
Ovom naredbom lokalne promenljive dobijaju svoje prethodne, globalne, vrednosti. Lokalne vrednosti se nepovratno gube.

Primer: Videti primer za naredbu **LOCAL.**

Izračunati GOTO

CGOTO

Namena: Bezuslovni prelazak na izvršavanje naredbi od izračunate programske linije.
Opšti oblik: CGOTO a

Argumenti: a — brojni izraz. Izraz čiji je rezultat brojna vrednost.

Ova naredba prevazilazi ograničenje standardne **GOTO** naredbe. Dozvoljava odlazak na programsku liniju čiji broj nije direktno zadat brojnom vrednošću, već je rezultat izračunavanja. To omogućuje višestruko grananje u programu bez višestruke upotrebe uslovnih naredbi.

Primer:

```
10 PRINT CHR$(147)
20 PRINT "1 - JEDAN IGRAC"
30 PRINT "2 - DVA IGRACA"
40 PRINT "0 - POCETAK IGRE"
50 PROC TASTER
60 : GET A$:IF A$<"0" OR A$>"2" THEN CALL TASTER
70 CGOTO 80+10*VAL(A$)
80 CALL POCETAK
90 EXEC JEDAN:GOTO 10
100 EXEC DVA:GOTO 10
```

Dati primer predstavlja početak jedne igre u kojoj igrač treba da pritisne jedan od odgovarajućih tastera (0, 1 ili 2) da bi izabrao broj igrača i otpočeo igru. U zavisnosti od pritisnutog tastera izračunava se na koju će se liniju otići, a time i na koju proceduru.

6.2.7 Rad sa brojevima

Aritmetičke operacije (MOD, DIV, FRAC, EXOR)

MOD

Namena: Dobijanje celobrojnog ostatka deljenja.
Opšti oblik: MOD (a, b)

Argumenti: a — broj koji se deli (deljenik). Mora biti ceo broj između 0 i 65535.
b — broj kojim se deli (delilac). Mora biti ceo broj između 0 i 65535.

Ovom funkcijskom naredbom dobija se ostatak deljenja dva broja. Na primer, deljenjem broja 7 sa 2 rezultat je 3, a ostatak 1. Ova naredba zamenjuje izraz **A-B * INT (A/B).**

Primer:	naredba	rezultat
	PRINT MOD (17,3)	**2**

DIV

Namena: Dobijanje celobrojnog rezultata delenja.
Opšti oblik: DIV (a, b)

Argumenti: a — broj koji se deli (deljenik). Mora biti ceo broj između 0 i 65535.
b — broj kojim se deli (delilac). Mora biti ceo broj između 0 i 65535.

Ova funkcijska naredba odgovara naredbi **INT (A/B).**

Primer:	naredba	rezultat
	PRINT DIV (17,3)	**5**

FRAC

Namena: Dobijanje decimalnog ostatka.
Opšti oblik: FRAC (a)
Argumenti: a — brojni izraz.

Ova funkcijska naredba odgovara naredbi **A-INT(A).**

Primer:	naredba	rezultat
	PRINT FRAC (−22/7)	**−.142857143**

EXOR

Namena: Izvršavanje logičkog isključivog sabiranja.
Opšti oblik: EXOR (a, b)

Argumenti: a – ceo broj između 0 i 65535.
b – ceo broj između 0 i 65535.

Ovom naredbom izvršava se operacija logičkog isključivog sabiranja nad bitima navedenih argumenata, po pravilima datim u opisu naredbe **WAIT** (standardni Komodorov bejzik).

Primer:	naredba	rezultat
	PRINT EXOR (5,3)	**6**

Rad sa binarnim i heksadecimalnim brojevima (%, $)

%

Namena: Pretvaranje binarnog broja u decimalni.
Opšti oblik: % bin

Argumenti: bin – binarni broj koji se sastoji od osam binarnih cifara.

Napomene: Ako iza znaka procenta nije navedeno tačno osam binarnih cifara prijaviće se izveštaj o grešci **NOT BINARY CHAR.** Ovo je funkcijska naredba.

Primer:
```
10 A=%00011001
15 PRINT A
```

$

Namena: Pretvaranje heksadecimalnog broja u decimalni.
Opšti oblik: $ hex

Argumenti: hex – četvorocifreni heksadecimalni broj.

Napomene: Ako iza znaka dolara nije navedeno tačno četiri heksadecimalne cifre prijaviće se izveštaj o grešci **NOT HEX CHAR.** Ovo je funkcijska naredba.
Primer: PRINT $00FF

6.2.8. Rad sa stringovima

Nove naredbe za rad sa stringovima (INSERT, INST, PLACE, DUP)

INSERT

Namena: Dobijanje novog stringa umetanjem jednog stringa u drugi.
Opšti oblik: INSERT („string 1", „string 2", n)

Argumenti: string 1 – string koji se umeće.
string 2 – string u koji se umeće.
n – redni broj karaktera u stringu 2 od koga se vrši umetanje.

Napomene: Argument n mora biti manji od dužine stringa 2. U protivnom prijaviće se izveštaj o grešci **?INSERT TOO LARGE.** Isti izveštaj će se dobiti ako je novo dobijeni string duži od 255 karaktera.

U direktnom načinu rada stringovi moraju biti dati u obliku promenljivih. U protivnom dobijeni string će biti nekorektan.

Sledećim primerom prikazano je dejstvo ove naredbe. Novi string je dobijen umetanjem jednog stringa u drugi.

Primer:
```
10 A$=INSERT(" ZA SVA","COMMODORE VREMENA",9)
20 PRINT A$
```

Novo dobijeni string je **COMMODORE ZA SVA VREMENA.** Sledeći primer prikazuje da stringovi mogu biti zadati i u obliku promenljivih.

Primer:
```
10 A$=" ZA SVA"
20 B$="COMMODORE VREMENA"
30 C$=INSERT (A$,B$,9)
40 PRINT C$
```

INST

Namena: Dobijanje novog stringa zamenom grupe karaktera jednog stringa drugim stringom.

Opšti oblik: INST („string 1", „string 2", n)

Argumenti: string 1 — string kojim se zamenjuje grupa karaktera u stringu 2.
string 2 — string u kome se zamenjuje grupa karaktera.
n — redni broj karaktera u stringu 2 od koga se vrši zamena.

Napomene: Argument n mora biti manji od dužine stringa 2. U protivnom neće doći do zamene dela stringa 2 stringom 1. U novodobijenom stringu pojaviće se neregularni karakteri. Dobijeni string ne sme biti duži od 255 karaktera.

U direktnom načinu rada stringovi moraju biti zadati u obliku promenljivih. Njihovo neposredno zadavanje rezultovaće nekorektnim stringom.

Sledeći primer ilustruje ovu naredbu u kojoj se podstring jednog stringa zamenjuje drugim stringom iste dužine.

Primer:
```
10 A$="XYZ"
20 B$="ABXDEFGHIJ"
30 FOR I=0 TO 10
40 : PPINT INST(A$,B$,I)
50 NEXT I
```

U programu stringovi mogu biti zadati i neposredno.

Primer:
```
10 A$=INST ("190","120 DIN.",0)
20 PRINT A$
```

PLACE

Namena: Nalaženje položaja podstringa u stringu.

Opšti oblik: PLACE („podstring", „string")

Argumenti: podstring — karakter ili string čiji se položaj traži.
string — string u kome se traži podstring.

Ova naredba omogućuje nalaženje jednog karaktera ili grupe karaktera u nekom stringu. Kao rezultat se dobija redni broj karaktera u stringu od koga počinje traženi string tj. podstring. Rezultat 0 označava da traženi podstring ne postoji.

U programskom načinu rada string i podstring mogu biti zadati i neposredno i u obliku promenljivih. U direktnom načinu rada mogu biti samo u obliku promenljivih. U protivnom će se dobiti pogrešan rezultat.

U sledećem primeru traži se položaj, tj. redni broj karaktera **K** u stringu **MIKRO**. Rezultat koji će se dobiti je 3.

Primer:
```
10 A$="MIKPO"
20 PRINT PLACE("K",A$)
```

DUP

Namena: Umnožavanje stringa.
Opšti oblik: DUP („string", n)

Argumenti: string — string koji se umnožava.
n — broj koji označava broj umnožavanja (n>0).

Napomena: Dobijeni string ne sme biti duži od 255 karaktera. U protivnom će rezultat biti pogrešan.

Ova naredba daje novi string koji je dobijen tako što se na polazni string nadovezuje taj isti string i to naznačenih n puta.

Primer: naredba — rezultat
PRINT DUP ("–", 12) — – – – – – – – – – – – –

Primer:
```
10 A$="YU"
20 FOR I=1 TO 10
30 : PRINT DUP (A$,I)
40 NEXT
```

6.2.9 **Ispisivanje rezultata**

Određivanje mesta ispisivanja i načina ispisivanja (CENTRE, USE, AT, LIN, PAUSE)

CENTRE

Namena: Ispisivanje stringa po sredini ekrana (centriranje).
Opšti oblik: CENTRE „string"

Argumenti: string — string koji se ispisuje. Njegova dužina ne sme biti veća od 39 karaktera.

Napomene: Po ispisivanju stringa kursor se, a time i novo ispisivanje ne prenosi u red niže.

Ovom naredbom se ostvaruje tzv. centriranje teksta. String se ispisuje tako da su njegov prvi i poslednji karakter podjednako udaljeni od leve, odnosno desne ivice ekrana. U slučaju stringa sa neparnim brojem karaktera, sa leve strane stringa ostavlja se jedno karakter polje više.

Primer:
```
10 FOR I-1 TO 21 STEP 2
20 : A$-DUP ("*",I)
30 : CENTRE A$:PRINT
40 NEXT
```

USE

Namena: Ispisivanje brojnih vrednosti u željenom formatu po težinskim vrednostima.
Opšti oblik: USE „string 1", „string 2"

Argumenti: string 1 — string koji određuje način ispisivanja. Može biti dat u obliku niza karaktera ili u obliku promenljive. U sebi može sadržati, za ovu naredbu kontrolni karakter # (povisilica).
string 2 — string koji se ispisuje. String sa numeričkom vrednošću.

Napomene: Naredbu treba koristiti samo u programskom načinu rada. U direktnom načinu će dati pogrešan rezultat.

Po ispisivanju stringa kursor, a time i novo ispisivanje se ne prenosi red niže.

Karakteri stringa 1 određuju ispisivanje karaktera stringa 2, i to tako što treba poklopiti decimalne tačke oba stringa. Na karakter mestima stringa 1, na kojima se nalazi znak povisilice biće preuzet karakter stringa 2. U slučaju da se u stringu ne nalazi decimalna tačka podrazumevaće se da je ona poslednji karakter u njemu. Na svim ostalim mestima biće preuzeti krakteri iz stringa 1. Na karakter mestima gde je kod stringa 1 povisilica, a kod stringa 2 ne postoji karakter biće upotrebljen karakter praznog polja.

Upotreba ove naredbe, koja je namenjena za određivanje načina ispisivanja decimalnih brojnih vrednosti, nalaže korišćenje naredbe za prevođenje brojeva u stringove (**STR$**).

Primer:
```
10 INPUT "UNESITE DECIMALNI BROJ";A
20 A$-STR$(A)
30 B$-"#"
40 FOR I-1 TO S
50 : USE "$ "+"##"+"."+C$,A$:PRINT
60 : C$-C$+B$
70 NEXT
```

U datom primeru uneti decimalni broj će se ispisati na pet načina. Jedinice i desetice će u svakom načinu biti ispisane. Njima će prethoditi znak dolara, S. Broj decimalnih mesta, desno od decimalne tačke će u svakom ispisivanju biti za po jedan veći. Sve to je određeno programskom linijom 50.

AT

Namena: Ispisivanje na željenom mestu ekrana.
Opšti oblik: PRINT AT (x, y) izraz
PRINT izraz 1 **AT** (x1, y1) izraz 2 **AT** (x2, y2) izraz 3...

Argumenti: x — broj kolone od koje će se ispisivati (od 0 do 39).
y — broj reda od koga će se ispisivati (od 0 do 24).
izraz — broj, brojna promenljiva, brojni izraz, karakter, string ili string izraz koji se ispisuje.

Napomene: Između reči **AT** i leve zagrade ne sme postojati prazno polje.

Ovom naredbom kursor se postavlja na željeno karakter polje ekrana. Ispisivanje koje sledi obaviće se od njega. Ako se želi postavljanje kursora na novo polje bez ispisivanja za izraz u naredbi treba navesti prazan string.

U sledećem primeru ispisuje se reč C64 na sredini ekrana.

Primer: PRINT AT (18, 12) „C64"

U sledećem primeru po ekranu se ispisuje 500 znakova * čiji se položaj bira slučajnim izborom.

Primer:
```
10 PRINT CHR$(147)
20 FOR I=1 TO 500
30    PRINTAT(40*RND(1),24*RND(1))"*"
40 NEXT
```

LIN

Namena: Određivanje broja reda ekrana u kome se nalazi kursor.
Opšti oblik: LIN

Ovom funkcijskom naredbom, uz standardnu Komodorovu naredbu za određivanje kolone zadnjeg ispisivanja **(POS)**, može se odrediti tačan položaj kursora na ekranu.

Primer:
```
10 PRINT CHR$(147)
20 A=20*RND(0)
30 FOR I=0 TO A
40 : PRINT "*"
50 NEXT
60 PRINT LIN
```

U datom primeru ispisuje se po jedna zvezdica u redu. Broj redova određuje generator slučajnih brojeva. Po završetku ispisivanja kursor silazi red niže, a pomoću naredbe **LIN** nalazi se i ispisuje red u kojem je.

PAUSE

Namena: Sporije izvršavanje programa.
Opšti oblik: PAUSE n
PAUSE „poruka", n

Argumenti: n — vreme, u sekundama, izvršenja naredbe **PAUSE** tj. čekanja do prelaska na sledeću naredbu.
poruka — bilo kakva poruka koja se ispisuje izvršenjem naredbe.

Napomene: Za zreme izvršavanja naredbe **PAUSE** pritiskom na taster **RETURN** prelazi se na sledeću naredbu tj. na dalje izvršavanje programa.

Za vreme izvršavanje naredbe **PAUSE** program nije moguće zaustaviti pritiskom na taster **RUN/STOP.** Pritisak na taster **RUN/STOP** i **RESTORE** i dalje zadržava svoje dejstvo.

Iza reči **PAUSE** i levog navodnika poruke obavezno je jedno prazno polje.

Izvršavanjem ove naredbe računar čeka da istekne vreme naznačeno argumentom n. Nakon toga prelazi na izvršavanje sledeće naredbe. Naredba se korisno može upotrebiti u toku razvoja programa, u cilju praćenja izvršavanja, a takođe i u gotovim programima za komotniji rad korisnika programa.

Primer: 10 PAUSE „OVO JE PAUZA OD 10 SEKUNDI", 10

6.2.10 **Boje na ekranu**

Treperenje ekrana (FLASH, OFF, BFLASH, BFLASH 0)

FLASH

Namena: Treperenje željene boje zapisa na ekranu.
Opšti oblik: FLASH n, t

Argumenti: n — broj koji označava boju koja treperi (od 0 do 15).
t — period treperenja (od 0 do 255).

Napomene: U visokoj rezoluciji (HIRES) i višebojnoj grafici (MULTI COLOUR) naredba nema uticaja.

Period treperenja raste u koracima od oko 1/60 sekunde za povećanje argumenta t po jedan. Za period od jedne sekunde t iznosi 58.

Treperenje je ostvareno tako što se zamenjuju boja zapisa (boja slova, brojeva, znakova...) i pozadine na kojoj je zapis ostvaren. Efekat je isti kao da se naizmenično obavlja inverzno i neinverzno **(RVS ON/RVS OFF)** ispisivanje.

Primer:

```
10 PRINT CHR$(147)
20 PRINT "(RED) COMMODORE(BLK)"
30 FOR I=10 TO 1 STEP-1
40 : FLASH 2,I
50 : PAUSE 1
60 NEXT
```

U datom primeru ispisuje se reč **COMMODORE** crvenom bojom. Daljim izvršavanjem programa ostvaruje se treperenje zamenjivanjem boje osnove i crvene boje zapisa što se daljim izvršavanjem programa obavlja sve brže i brže.

OFF

Namena: Zaustavljanje treperenja boja na ekranu.
Opšti oblik: OFF

Primer: OFF

Nakon izvršenja primera za naredbu **FLASH** direktnim zadavanjem naredbe **OFF** prestaće treperenja boja na ekranu.

BFLASH

Namena: Treperenje u boji okvirnog dela ekrana.
Opšti oblik: BFLASH t, n, m

Argumenti: t — period treperenja ($1<t<255$).
n, m — brojevi boja koje se menjaju na okviru ekrana ($0\leq n\leq 15$), ($0\leq m\leq 15$).

Napomene: Period treperenja raste u koracima od oko 1/60 dela sekundi za povećanje argumenta t za po jedan (t=58 za period od jedne sekunde).

Primer: BFLASH 30,6,7

Okvirni deo ekrana menjaće boju iz plave u žutu, i obrnuto svakih pola sekundi.

BFLASH 0

Namena: Prestanak treperenja boja na okvirnom delu ekrana.
Opšti oblik: BFLASH 0

Primer: BFLASH 0

Po izvršenju primera za naredbu **BFLASH** izvršiti naredbu **BFLASH 0.**

Postavljanje boja na ekran (COLOUR, BCKGNDS)

COLOUR

Namena: Postavljanje boje okvira ekrana i boje pozadine središnjeg dela ekrana.
Opšti oblik: COLOUR n, m

Argumenti: n – broj od 0 do 15 koji označava boju okvirnog dela ekrana.
m – broj od 0 do 15 koji označava boju središnjeg dela ekrana.

Primer:

```
10 FOR I=0 TO 15
20 : FOR J=0 TO 15
30 :   COLOUR I,J
40 :   PAUSE 1
50 : NEXT
60 NEXT
```

Datim programom smenjuje se svih 256 mogućih kombinacija za 16 mogućih boja okvira i 16 mogućih boja središnjeg dela ekrana. Prolazak kroz program može se ubrzati pritiskom na taster RETURN.

BCKGNDS

Namena: Promena boje pozadine karaktera.
Opšti oblik: BCKGNDS a, b, c, d

Argumenti: a – broj od 0 do 15 koji određuje boju koja se dodeljuje svim karakterima sa ekranskim kodovima od 0 do 63
b – broj od 0 do 15 koji određuje boju koja se dodeljuje svim karakterima sa ekranskim kodovima od 64 do 127.
c – broj od 0 do 15 koji određuje boju koja se dodeljuje svim karakterima sa ekranskim kodovima od 128 do 191.
d – broj od 0 do 15 koji određuje boju koja se dodeljuje svim karakterima sa ekranskim kodovima od 192 do 255.

Napomene: Moguće je dobiti samo prvih 64 karaktera na četiri boje pozadine.
Povratak na normalan način ispisivanja ostvaruje se naredbom **NRM.**

Ovom naredbom se prelazi u tekstualni rad sa višebojnom pozadinom (EXTENDED COLOUR).

Izvršavanjem naredbe pozadina će postati one boje koja je određena argumentom a. Držanjem pritisnutog tastera **SHIFT** i pritiskanjem tastera, karakteri će se ispisivati sa pozadinom određenom argumentom b. Prelaskom u inverzni način ispisivanja **(CTRL 9)** karakteri će se ispisivati sa pozadinom određenom argumentom c. Držanjem pritisnutog tastera **SHIFT** u inverznom načinu ispisivanja dobiće se pozadina određena argumentom d.

Primer: BCKGNDS 1,3,5,7

Izvršenjem navedene naredbe karakteri će se ispisivati na beloj pozadini. Uz pritisnut taster SHIFT na svetlo plavoj, u inverznom načinu na zelenoj, a ako je još pritisnut i taster SHIFT na žutoj osnovi.

Popunjavanje ekrana tekstualnim znacima i inverzno ispisivanje (FCHR, FCOL, FILL, MOVE, INV)

FCHR

Namena: Ispisivanje jednog karaktera u naznačeno pravougaono polje na ekranu.
Opšti oblik: FCHR y,x,b,a,n

Argumenti: y — broj reda u kome se nalazi gornja stranica pravougaonog polja (od 0 do 24).
x — broj kolone u kojoj se nalazi leva stranica pravougaonog polja (od 0 do 39).
b — visina pravougaonog polja u karakterima.
a — širina pravougaonog polja u karakterima.
n — ekranski kôd karaktera.

Napomene: Zbir argumenata y+b mora biti manji od 25, a zbir argumenata x+a manji od 40.

Ovom naredbom ostvaruje se ispisivanje jednog željenog karaktera na svim karakter poljima ekrana unutar pravougaonog polja čije se dimenzije zadaju. Zadavanje karaktera ostvaruje se pomoću ekranskih kodova. Za razliku od ASCII kodova, ekranski kodovi ne uključuju kontrolne kodove. Ograničeni su na znakove koji se ispisuju na ekranu i za iste vrednosti ne daju iste karaktere kao ASCII kodovi. Njihov spisak je dat na kraju knjige u tabeli kodova.

Primer: FCHR 10,20,8,10,1

Datim primerom u pravougaono polje dimenzija 8×10, čiji se levi gornji ugao nalazi u desetom redu i dvadesetoj koloni ispisuje se ukupno 80 slova A.

FCOL

Namena: Promena boje zapisa unutar pravougaonog polja na ekranu.
Opšti oblik: FCOL y,x,a,b,n

Argumenti: y — red u kome se nalazi gornja stranica pravougaonog polja (od 0 do 24).
x — kolona u kojoj se nalazi leva stranica pravougaonog polja (od 0 do 39).
a — širina pravougaonog polja u karakterima.
b — visina pravougaonog polja u karakterima.
n — broj (kôd) boje u koju će preći zapis unutar pravougaonog polja (od 0 do 15).

Napomene: Zbir argumenata y+b mora biti manji od 25, a zbir argumenata x+a manji od 40.

Ovom naredbom se menja samo boja zapisa, a ne i boja osnove unutar naznačenog pravougaonika (npr. crveno ispisano slovo A na žutoj osnovi postaje zeleno ispisano takođe na žutoj osnovi).

Primer:

```
10 PRINT CHR$(147)
20 FCHR 0,0,20,18,1
30 FOR Y-0 TO 15
40 : FCOL Y+1,1,18,1,Y
50 NEXT
60 PAUSE 10
```

FILL

Namena: Ispisivanje jednog karaktera naznačene boje u naznačeno pravougaono polje na ekranu.

Opšti oblik: **FILL** y,x,a,b,n,m

Argumenti:
- y — broj reda u kome se nalazi gornja stranica pravougaonog polja (od 0 do 24).
- x — broj kolone u kojoj se nalazi leva stranica pravougaonog polja (od 0 do 39).
- a — širina pravougaonog polja u karakterima.
- b — visina pravougaonog polja u karakterima.
- n — ekranski kôd karaktera.
- m — broj koji određuje boju karaktera.

Napomene: Zbir argumenata y+b mora biti manji od 25, a zbir argumenata x+a manji od 40.

Za razliku od naredbe **FCHR** koja zadato polje popunjava zadatim karakterom u tekućoj boji ispisivanja, ovom naredbom se zadaje i boja ispisivanja karaktera.

Ovom naredbom ostvaruje se ispisivanje jednog željenog karaktera, u jednoj željenoj boji, na svim karakter poljima ekrana unutar pravougaonog polja čije se dimenzije zadaju. Zadavanje karaktera ostvaruje se pomoću njihovih ekranskih kodova.

Primer: **FILL 10,20,8,10,1,1**

Datim primerom u pravougaono polje dimenzija 8×10, čiji se levi gornji ugao nalazi u desetom redu i dvadesetoj koloni ispisuje se ukupno 80 slova A belom bojom.

MOVE

Namena: Kopiranje pravougaonog polja ekrana.

Opšti oblik: **MOVE** y, x, a, b, y1, x1

Argumenti:
- y — red u kome se nalazi gornja stranica pravougaonog polja (od 0 do 24).
- x — kolona u kojoj se nalazi leva stranica pravougaonog polja (od 0 do 39).
- a — širina pravougaonog polja u karakterima.
- b — visina pravougaonog polja u karakterima.
- y1 — red u kome će se nalaziti gornja stranica kopije pravougaonog polja.
- x1 — kolona u kojoj će se nalaziti leva stranica kopije pravougaonog polja.

Napomene: Zbir argumenata y + b mora biti manji od 25, a zbir argumenata x +a manji od 40. To isto važi i za zbirove y1 +b i x1 +a respektivno.

Ovom naredbom se praktično izvršava dupliciranje onoga što je ispisano na jednom delu ekrana, na neki drugi deo ekrana.

Primer:

```
10 PRINT CHR$(147)
20 FCHR 5,2,6,8,1
30 PAUSE 1
40 MOVE 5,2,6,8,10,20
```

INV

Namena: Zamena boje teksta i pozadine u pravougaonom polju na ekranu.

Opšti oblik: **INV** y, x, a, b

Argumenti: y – red u kome se nalazi gornja stranica pravougaonog polja (od 0 do 24).
x – kolona u kojoj se nalazi leva stranica pravougaonog polja (od 0 do 39).
a – širina pravougaonog polja u karakterima.
b – visina pravougaonog polja u karakterima.

Napomene: Zbir argumenata y+b mora biti manji od 24, a argumenata x+a manji od 39.

Ovom naredbom se ostvaruje da ono što je ispisano, na nekom delu ekrana, bude ispisano bojom osnove, a pri tome osnova dobija boju zapisa. To je takozvani inverzni način ispisivanja.

Primer:
```
10 PRINT CHR$(147)
20 FCHR 5,2,6,8,1
30 PAUSE 1
40 INV 5,2,6,8
```

6.2.11 Pomeranje sadržaja ekrana

(LEFT, RIGHT, UP, DOWN)

LEFT

Namena: Pomeranje karaktera u levo za jedno karakter mesto unutar pravougaonog polja na ekranu (engl. scroll left).

Opšti oblik: **LEFTB** y, x, a, b
LEFTW y, x, a, b

Argumenti: y – Red u kome se nalazi gornja stranica pravougaonog polja (od 0 do 24).
x – Kolona u kojoj se nalazi leva stranica pravougaonog polja (od 0 do 39).
a – Širina pravougaonog polja u karakterima.
b – Visina pravougaonog polja u karakterima.

Napomene: Ako je u imenu naredbe kao zadnje slovo navedeno B, prilikom pomeranja izgubiće se red sa one strane na koju se vrši pomeranje. Ako je navedeno W doći će do pomeranja karaktera „u krug". Prvi karakter se pojavljuje kao poslednji.

Zbir argumenata y+b mora biti manji od 25, a argumenata x+a manji od 40.

Primer:
```
10 PRINT CHR$(147)
20 FOR I=1 TO 8
30 : PRINT TAB(3);"ABC 123"
40 NEXT
50 FOR I=1 TO 7
60 : PAUSE 1
70 : LEFTB 4,3,7,6
80 NEXT
```

Dati primer prikazuje naredbu **LEFTB.** Isti primer može poslužiti i za prikazivanje naredbe **LEFTW,** ako se u liniji 70 zameni naredba **LEFTB** sa **LEFTW.**

RIGHT

Namena: Pomeranje karaktera u desno za jedno karakter mesto unutar pravougaonog polja na ekranu (engl. scroll right).

Opšti oblik: **RIGHTB** y, x, a, b
RIGHTW y, x, a, b

Argumenti: y, x, a, b – identično kao kod naredbe **LEFT**.

Napomene: Identično kao kod naredbe **LEFT**.

Primer: Upotrebiti primer za naredbu **LEFT**. U liniji 70 zameniti naredbu **LEFTB** prvo sa **RIGHTB**, pa sa **RIGHTW**.

UP

Namena: Pomeranje karaktera na gore za jedno karakter mesto unutar pravougaonog polja na ekranu (engl. scroll up).

Opšti oblik: **UPB** y, x, a, b
UPW y, x, a, b

Argumenti: y, x, a, b – identično kao kod naredbe **LEFT**.

Napomene: Identično kao kod naredbe **LEFT**.

Primer:

```
10 PRINT CHR$(147)
20 FOR I=1 TO 8
30 : PRINT TAB(3);DUP("*",I)
40 NEXT
50 FOR I=1 TO 8
60 : PAUSE 1
70 : UPB 1,3,5,8
80 NEXT
```

Dati primer prikazuje naredbu **UPB**. Isti primer može poslužiti i za prikazivanje naredbe **UPW**, ako se u liniji 70 zameni naredba **UPB** sa **UPW**.

DOWN

Namena: Pomeranje karaktera na dole za jedno karakter mesto unutar pravougaonog polja na ekranu (engl. scroll down).

Opšti oblik: **DOWNB** y, x, a, b
DOWNW y, x, a, b

Argumenti: y, x, a, b – identično kao kod naredbe **LEFT**.

Napomene: Identično kao kod naredbe **LEFT**.

Primer: Upotrebiti primer za naredbu **UP**. U liniji 70 zameniti naredbu **UPB** prvo sa **DOWNB**, pa sa **DOWNW**.

6.2.12 Grafika

Zadavanje grafičkog načina rada (HIRES, MULTI, NRM, CSET, LOW COL, HI COL)

HIRES

Namena: Prelazak u HRG način rada (grafika visoke rezolucije) i zadavanje boje zapisa i boje pozadine.

Opšti oblik: **HIRES** a, b

Argumenti: a — broj koji određuje boju zapisa (tačaka). Boje i njima odgovarajuće brojne vrednosti date su u poglavlju o grafici.
b — broj koji određuje boju pozadine.

Napomena: Po prestanku izvršavanja programa Komodor ispisuje neku od poruka (**READY, STOP...**) i pri tome prelazi u tekstualni način rada. Sadržaj visoko rezolucijskog ekrana, koji se pri tome briše, može se povratiti naredbom **CSET 2.**

U visokoj rezoluciji svakoj tački ekrana odgovara jedan bit HRG memorije. Ukupno ih ima 320 (po horizontali) puta 200 (po vertikali). Ovom naredbom briše se prethodni sadržaj visoko rezolucijskog ekrana, odnosno sadržaj HRG memorije. Nakon toga mogu se primenjivati sve odgovarajuće naredbe za rad u visokoj rezoluciji, odnosno naredbe za grafiku.

Prelaskom u HRG način rada, prethodni, tekstualni sadržaj ekrana se gubi, ali biva povraćen nazad, povratkom u tekstualni način rada.

Primer:

```
10 FOR A=0 TO 14
20 HIRES A,15
30 : FOR X=59968 TO 60159
40 :  POKE X,%11001100
50 : NEXT
60 NEXT
```

U datom primeru, na sivoj osnovi ispisuje se redom u svim bojama odgovarajuća šara. Šara je dobijena direktnim ubacivanjem vrednosti u HRG memoriju (linija 40).

MULTI

Namena: Prelazak u višebojni način rada i zadavanje boja.
Opšti oblik: **MULTI** l, d, z

Argumenti: l — broj koji određuje boju leve ivice zapisa.
d — broj koji određuje boju desne ivice zapisa.
z — broj koji određuje boju samog zapisa.
Vrednosti svih argumenata mogu biti od 0 do 15.

Napomene: Za povratak iz višebojnog načina rada koristi se naredba NRM.

Ova naredba koristi se za prelazak u višebojni način rada (MULTI COLOUR). Time se unutar polja ekrana, koje odgovara jednom karakteru mogu dobiti četiri boje.

Za razliku od rada u visokoj rezoluciji u kojoj je slika na ekranu sačinjena od 320 puta 200 tačaka, u višebojnom načinu rada slika je sačinjena od 160 (po horizontali) puta 200 tačaka (po vertikali). Po horizontali je upola manje tačaka, ali kako je širina slike ista, može se reći da su tačke dva puta šire. Praktično, tačka u višebojnom načinu rada je sastavljena od dve tačke koje se pojavljuju u visokoj rezoluciji.

Svakoj tački u visokoj rezoluciji odgovara jedan bit visoko rezolucijske (HRG) video memorije. Za višebojni način rada to su dva bita. Dva bita mogu dati četiri kombinacije vrednosti kojima odgovaraju sledeće boje:

vrednosti

00 — Višebojna tačka kojoj odgovaraju ove vrednosti imaće boju prethodno zadate pozadine. Pri tome svi karakteri imaju istu boju pozadine.

01 — Višebojna tačka kojoj odgovaraju ove vrednosti imaće boju zadatu argumentom 1. Vrednost 01 odgovara levoj tački nevidljivoj (0 — iste boje kao pozadina), a

desnoj tački vidljivoj (1 — boje zapisa). Praktično, radi se o desnoj ivici nečega ispisanog (zapisa) na ekranu.

10 — Višebojna tačka kojoj odgovaraju ove vrednosti imaće boju zadatu argumentom d. Vrednosti 10 odgovaraju desnoj tački nevidljivoj (0), a levoj tački vidljivoj (1). Radi se o desnoj ivici zapisa.

11 — Višebojna tačka kojoj odgovaraju ove vrednosti imaće boju zadatu argumentom z. To je upravo boja zapisa, boja u koju će preći sve što je ispisano pri povratku iz višebojnog načina rada.

Primer:

```
10 HIRES 0,2
20 FOR I=0 TO 25
30 : A=50-I
40 : LINE I,25,I+25,0,1
50 NEXT
60 MULTI 1,5,8
70 GOTO 70
```

Linijom 10 prelazi se u visoko rezolucijski način rada sa ispisivanjem u crnoj boji na crvenoj osnovi. Petljom je ostvareno iscrtavanje romba. Naredbom u liniji 60 prelazi se u višebojni način rada. Leva ivica (prelazak sa crvenog na crno) postaje bele boje (1), desna ivica (prelazak sa crnog na crveno) postaje zelene boje (5), a sam zapis (crne boje) postaje narandžast (8).

Ovom naredbom ne može se u potpunosti ostvariti višebojni tekstualni način rada. Može se postaviti samo boja zapisa (argument z). Pri tome se mogu dobiti boje zapisa od crnog do žutog. Vrednost 8 odgovara crnoj boji, a 15 žutoj. Za druge vrednosti neće se dobiti višebojni karakter. Boja levih ivica (svih karaktera) i desnih ivica (svih karaktera) može se postaviti direktnim upisivanjem vrednosti boja u memorijske lokacije 53282 odnosno 53283. Za dalja objašnjenja videti poglavlje 9.

NRM

Namena: Povratak u tekstualni način rada.
Opšti oblik: NRM

Napomene: Po prestanku izvršavanja programa, i ispisivanju poruke računar automatski prelazi u tekstualni način rada.

Ovom naredbom obavlja se povratak iz visoko rezolucijskog načina rada (HRG), višebojnog načina rada (MULTI COLOUR) ili iz načina rada sa višebojnom pozadinom (EXTENDED BACKGROUND). Ponovo se dobija prethodni tekstualni sadržaj ekrana.

Primer: U primeru za naredbu **MULTI** ubaciti liniju:
65 NRM

CSET

Namena: Promena sadržaja ekrana iz tekstualnog u grafički i obrnuto.
Opšti oblik: CSET n

Argumenti: n=0 prelazak na tekstualni sadržaj ekrana (osnovni set karkatera tj. velika slova).
1 prelazak na tekstualni sadržaj ekrana (drugi set karaktera tj. mala slova).
2 prelazak u grafiku visoke rezolucije.

Ovom naredbom, sem što se može promeniti set karaktera, može se povratiti sadržaj visoko rezolucijskog ekrana, kako osnovnog (HRG) tako i višebojnog (MULTI COLOUR).

Primer:
```
10 HIRES 1,2
20 FOR I=0 TO 25
30 : REC I,I,50-I,50-I,1
40 NEXT
50 PAUSE 3:CSET 1
60 IF INKEY=1 THEN CSET 2:PAUSE 4:END
70 GOTO 60
```

U datom primeru se nakon iscrtavanja kvadrata u visokoj rezoluciji ostvaruje povratak tekstualnog sadržaja ekrana, ali u drugom setu karaktera (linija 50). Pritiskom na funkcijski taster f1, ponovo se dobija sadržaj visoko rezolucijskog ekrana (linija 60).

LOW COL

Namena: Zadavanje boja unutar svakog karakter polja ekrana.
Opšti oblik: LOW COL l, d, z

Argumenti: U višebojnom načinu rada (MULTI COLOUR):
- l — boja leve ivice zapisa (kao u naredbi **MULTI**)
- d — boja desne ivice zapisa (kao u naredbi **MULTI**)
- z — boja zapisa.

U normalnoj visokoj rezoluciji (HRG):
- l — boja zapisa.
- d — boja osnove.
- z — bez uticaja na rad.

Ovom naredbom se menjaju boje koje se mogu pojaviti unutar polja na ekranu koje odgovara jednom karakteru. Naredbu je neophodno zadati pre neke od naredbi za crtanje. Iscrtavanjem menjaju se odgovarajuće boje unutar karakter polja. Ova naredba nadopunjuje naredbe **HIRES** i **MULTI** koje su zadavale boje za sva karakter polja ekrana zajedno.

Primer:
```
10 HIRES 0,1
20 FOR I=0 TO 15
30 : LOW COL I,2,0
40 : LINE 0,8*I+1,200+5*I,8*I+1,1
50 NEXT
60 GOTO 60
```

Dati primer prikazuje upotrebu ove naredbe u crtanju u visokoj rezoluciji (HRG). Boja kojom se crta menja se sa svakom linijom kroz sve boje. Boja osnove karakter polja kroz koje prolazi linija prelazi u crvenu (argument 2 u liniji 30). Treći argument nema uticaja na rad.

HI COL

Namena: Prekid dejstva naredbe **LOW COL**. Povratak na boje zadate naredbama **HIRES** odnosno **MULTI**.
Opšti oblik: HI COL

Upotrebom ove naredbe dalja iscrtavanja neće se obavijati bojama zadatim naredbom **HI COL**, već prethodno zadatim bojama naredbama **HIRES** odnosno **MULTI**.

Primer: U primeru za naredbu **LOW COL** dodati linije.

```
45 HI COL
46 LINE 0,8*I+3,300,8*I+3,1
```

Obratiti pažnju da se iscrtavanje u prethodnim bojama obavlja samo u karakter poljima kojima boje već nisu postavljene naredbom **HI COL**.

Crtanje i bojenje (PLOT, TEST, LINE, REC, BLOCK, CIRCLE, ARC, ANGL, PAINT, DRAW, ROT)

PLOT

Namena: Crtanje jedne tačke na ekranu.

Opšti oblik: **PLOT** x, y, n

Argumenti: x — x koordinata tačke (od 0 do 319 za HRG, a od 0 do 159 za MULTI COLOUR).

y — y koordinata tačke (od 0 do 199).

n — broj koji označava način crtanja.

U radu u visokoj rezoluciji (HRG) ima sledeća značenja:

n = 0 Brisanje tačaka.

n = 1 Crtanje tačaka.

n = 2 Invertovanje tačaka (nacrtane tačke se brišu, a nenacrtane se crtaju)

U višebojnom načinu rada (MULTI COLOUR) ima sledeća značenja:

n = 0 Brisanje tačaka.

n = 1 Crtanje tačaka tačkama i bojom leve ivice (par 01).

n = 2 Crtanje tačaka tačkama i bojom desne ivice (par 10).

n = 3 Invertovanje tačaka (parovi 01 postaju 10 i obrnuto, a zapis tj. par 11 zamenjuje se sa pozadinom tj. sa parom 00 i obrnuto).

Ovom naredbom se crta ili briše tačka na zadatim kordinatama x i y. x kordinata označava njenu udaljenost od leve strane središnjeg dela (pozadine) ekrana po kome se crta, a y kordinata označava udaljenost od gornje strane središnjeg dela ekrana (koordinatni početak je u levom gornjem uglu). Horizontalu središnjeg dela ekrana sačinjavaju 320 tačaka označenih od 0 do 319, a vertikalu 200 tačaka označenih od 0 do 199. U višebojnom načinu rada (MULTI COLOUR) pozadina se po horizontali sastoji od 160 tačaka označenih od 0 do 159.

Primer:

```
10 HIRES 0,1
20 REPEAT
30 :  N=N+1
40 :  FOR I=0 TO 319
50 :    PLOT I,100-90*SIN(I/30),N
60 :  NEXT
70 :  PAUSE 2
80 UNTIL N=2
```

TEST

Namena: Utvrđivanje da li je na navedenim koordinatama nacrtana tačka.

Opšti oblik: **TEST** (x, y)

Argumenti: x — x koordinata tačke (identično kao u naredbi **PLOT**).

y — y koordinata tačke (identično kao u naredbi **PLOT**).

Napomena: Ovo je funkcijska naredba.

Primer: Nakon izvršenja primera za naredbu PLOT zadati naredbe:
PRINT TEST (0,0) i
PRINT TEST (0,100)

LINE

Namena: Crtanje prave linije na ekranu.
Opšti oblik: **LINE** x, y, x1, y1, n

Argumenti: x — x koordinata početne tačke linije.
y — y koordinata početne tačke linije.
x1 — x koordinata krajnje tačke linije.
y1 — y koordinata krajnje tačke linije.
n — broj koji označava način crtanja.
Sve navedeno za argumente u naredbi **PLOT** važi i u ovoj naredbi.

Primer:

```
10 HIRES 1,0
20 FOR I=0 TO 319 STEP 5
30 : LINE 0,0,I,199,1
40 NEXT
50 FOR I=0 TO 319 STEP 5
60 : LINE 319,199,I,0,1
70 NEXT
80 GOTO 80
```

REC

Namena: Crtanje pravougaonika.
Opšti oblik: **REC** x, y, a, b, n

Argumenti: x — x koordinata gornjeg levog ugla pravougaonika.
y — y koordinata gornjeg levog ugla pravougaonika.
a — širina pravougaonika.
b — visina pravougaonika.
n — način crtanja.
Sve navedeno za argumente u naredbi **PLOT** važi i u ovoj naredbi.

Napomene: Zbir argumenata x+a treba biti manji od 320 za HRG način rada, odnosno manji od 160 za višebojni način rada (MULTI COLOUR). Zbir argumenata y+b treba biti manji od 200.

Primer:

```
10 COLOUR 2,2
20 HIRES 0,1
30 FOR I=0 TO 80 STEP 3
40 : REC I,I,I,I,1
50 NEXT
60 MULTI 1,5,0
70 GOTO 70
```

BLOCK

Namena: Crtanje popunjenog pravougaonika.
Opšti oblik: **BLOCK** x, y, x1, y1, n

Argumenti: x – x koordinata gornjeg levog ugla pravougaonika.
y – y koordinata gornjeg levog ugla pravougaonika.
x1 – x koordinata donjeg desnog ugla pravougaonika.
y1 – y koordinata donjeg desnog ugla pravougaonika.
n – način crtanja.
Sve ostalo navedeno za argumente u naredbi **PLOT** važi i u ovoj naredbi.

Napomene: x treba biti manje od x1, a y manje od y1.

Ovom naredbom se, za razliku od naredbe **REC** kojom se crtaju samo stranice pravougaonika, crta potpuno popunjen pravougaonik.

Primer:

```
10 COLOUR 1,1:HIRES 6,1
20 FOR I=1 TO 15
30 : X=159*RND(0):Y=199*RND(0)
40 : X1=159*RND(0):Y1=199*RNO(0)
50 : IF X>X1 THEN A=X:X=X1:X1=A
60 : IF Y>Y1 THEN A=Y:Y=Y1:Y1=A
70 : BLOCK X,Y,X1,Y1,2
80 NEXT
90 GOTO 90
```

U datom primeru iscrtava se u levoj polovini ekrana 15 plavo popunjenih pravougaonika na beloj osnovi. Iscrtavanje je određeno petim argumentom u naredbi **BLOCK.** Za vrednost 2 pravougaonici se iscrtavaju inverzno (u datom slučaju na beloj pozadini crtaju se plavo, a na plavoj pozadini belo).

CIRCLE

Namena: Crtanje kružnice i elipse.
Opšti oblik: **CIRCLE** x, y, a, b, n

Argumenti: x – x koordinata centra kružnice tj. elipse.
y – y koordinata centra kružnice tj. elipse.
a – poluose elipse u x pravcu tj. poluprečnik kružnice.
b – poluosa elipse u y pravcu tj. poluprečnik kružnice,
n – način crtanja.
Sve ostalo navedeno za argumente u naredbi **PLOT** važi i u ovoj naredbi.

Napomene: Da bi se na ekranu nacrtao krug mora biti zadovoljeno sledeće:

x poluosa=1.15*y poluosa (za HRG)
y poluosa=0.575*y poluosa (za MULTI COLOUR)
U svakom slučaju treba biti zadovoljeno sledeće:
x>=a, y>=b
y+b<=199 (za HRG i MULTI COLOUR)
x+a<=319 (za HRG), y+b<=159 (za MULTI COLOUR)

Primer:

```
10 COLOUR 7,7:HIRES 0,7
20 FOR I=1 TO 10
30 : X=80+160*RND(0):Y=50+100*RNO(0)
40 : A=80*RNO(0):B=50*RNO(0)
50 : CIRCLE X,Y,A,B,1
60 NEXT
70 GOTO 70
```

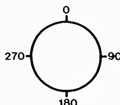
U datom primeru iscrtava se 10 elipsi crnom bojom na žutoj pozadini. Može se uočiti da poluose elipsi mogu biti samo u pravcu x i y ose.

ARC

Namena: Crtanje dela kružnice ili elipse.
Opšti oblik: **ARC** x, y, u1, u2, g, a, b, n

Argumenti: x – x koordinata centra kružnice tj. elipse.
y – y koordinata centra kružnice tj. elipse.
u1 – početni ugao.
u2 – krajnji ugao.
g – ugao između pravih od kojih je sastavljen luk kružnice ili elipse.
a – poluprečnik kružnice tj. poluosa elipse u x pravcu.
b – poluprečnik kružnice tj. poluosa elipse u y pravcu.
n – način crtanja.
Sve ostalo navedeno za argumente u naredbi **PLOT** važi i u ovoj naredbi.

Napomene: Uglove treba navesti u stepenima.
Smer porasta i orijentacija uglova data je na sledećem crtežu:



Primer:

```
10 HIRES 5,7:COLOUR 7,7
20 FOR I=1 TO 10
30 : ARC 160,100,0,36*I,5*I,10*I,10*I,1
40 NEXT
50 GOTO 50
```

U datom primeru iscrtava se 10 kružnih lukova. U svakom narednom luku povećava se ugao koji luk opisuje i poluprečnik kružnice kome luk pripada. Pažnju treba obratiti da se povećava i argument g što dovodi do toga da je luk sastavljen iz sve manjeg broja sve dužih pravih linija. Uočava se da vrednost 1 za argument g daje liniju najbližu luku. Vrednost 0 daće samo tačku sa najmanjom y koordinatom. (nije prikazano u primeru).

ANGL

Namena: Crtanje poluprečnika kružnice i poluose elipse.
Opšti oblik: **ANGL** x, y, u, a, b, n

Argumenti: x – x koordinata centra kružnice tj. elipse.
y – y koordinata centra kružnice tj. elipse.
u – ugao unutar koga se vrši iscrtavanje.
a – poluprečnik kružnice tj. poluosa elipse u x pravcu.
b – poluprečnik kružnice tj. poluosa elipse u y pravcu.
n – način crtanja.
Sve ostalo navedeno za argumente u naredbi **PLOT** važi i u ovoj naredbi.

Napomene: Identično kao u naredbi **ARC.**

Primer:
```
10 HIRES 5,7:COLOUR 5,7
20 FOR I=1 TO 30
30 : ANGL 160,100,10*I,5*I,5*I,1
40 NEXT
50 GOTO 50
```

PAINT

Namena: Popunjavanje površine unutar zatvorene linije.

Opšti oblik: **PAINT** x, y, n

Argumenti: x — x koordinata tačke koja treba biti unutar zatvorene linije.
y — y koordinata tačke koja treba biti unutar zatvorene linije.
n — način popunjavanja.
Sve ostalo navedeno za argumente u naredbi PLOT važi i u ovoj naredbi.

Primer:
```
10 HIRES 0,7:COLOUR 7,7
20 CIRCLE 130,100,50,50,1
30 CIRCLE 190,100,40,30,1
40 PAINT 160,100,1
50 GOTO 50
```

DRAW

Namena: Crtanje proizvoljnog geometrijskog lika.

Opšti oblik: **DRAW** „string", x, y, n

Argumenti: string — string koji se sastoji od brojeva.
x — x koordinata početne tačke.
y — y koordinata krajnje tačke.
n — način crtanja.
Sve ostalo navedeno za argumente u naredbi **PLOT** važi i u ovoj naredbi.

U brojnom stringu svaka cifra određuje položaj tačke u odnosu na prethodnu tačku, a takođe i način njenog crtanja. Vrednosti, položaji i načini crtanja su sledeći:

0	desno	bez ispisivanja
2	dole	bez ispisivanja
3	levo	bez ispisivanja
4	identično kao za vrednost 2	
5	desno	crtanje jedne tačke
6	gore	crtanje jedne tačke
7	dole	crtanje jedne tačke
8	levo	crtanje jedne tačke
9	prestanak crtanja	

Primer:
```
10 HIRES 0,1
20 A$="555555555":B$="666666666"
30 C$="888888888":D$="777777777"
40 FOR I=0 TO 6
50 : E$=A$+B$+C$+D$
60 : ROT I,I+1
70 : DRAW E$,150,110,1
80 : PAUSE 1
90 NEXT
100 GOTO 100
```

U datom primeru iscrtava se 7 kvadrata sa stranicama određenim vrednostima string promenljivih. Položaj i veličina kvadrata određena je naredbom **ROT** u liniji 60.

ROT

Namena: Zadavanje položaja rotacije i veličine crteža nacrtanog naredbom **DRAW.**

Opšti oblik: **ROT** r, a

Argumenti: r — broj koji određuje ugao zakretanja u odnosu na osnovni položaj. Porast ugla je u smeru kazaljke na satu. Date su dozvoljene vrednosti argumenta i njima odgovarajući uglovi:

r	ugao zakretanja
0	0
1	45
2	90
3	135
4	180
5	225
6	270
7	315

a — broj koji svojom vrednošću direktno određuje povećanje crteža. Mora biti između 1 i 255. Vrednost 1 odgovara crtežu bez povećanja.

Primer: Videti primer za naredbu **DRAW.**

Kombinovanje teksta sa grafikom (CHAR, TEXT)

CHAR

Namena: Ispisivanje karaktera na visoko rezolucijskom (HRG) i višebojnom (MULTI COLOUR) ekranu.

Opšti oblik: **CHAR** x, y, k, n, a

Argumenti:
- x — x koordinata leve ivice karaktera.
- y — y koordinata gornje ivice karaktera.
- k — ekranski kôd karaktera.
- n — način crtanja.
- a — povećanje karaktera po vertikali. Za vrednosti 0 i 1 karakter se ispisuje bez povećanja.

Sve ostalo navedeno za argumente u naredbi **PLOT** važi i u ovoj naredbi.

Primer:
```
10 HIRES 3,1
20 FOR I=1 TO 26
30 : CHAR 10*I,10,I,1,I
40 NEXT
50 GOTO 50
```

TEXT

Namena: Ispisivanje niza karaktera na visoko rezolucijskom (HRG) i višebojnom (MULTI COLOUR) ekranu.

a=57344+8*k ili a=$E000+8*k za prvi set
a=59392+8*k ili a=$E800+8*k za drugi set

gde je k ekranski kôd karaktera.

Primer:

```
10 MEM
20 DESIGN 2,57344+8*1
100 @........
110 @........
120 @...B..B.
130 @...B..B.
140 @...B..B.
150 @..B.BB.B
160 @.B......
170 @B.......
```

U datom primeru definiše se grčko slovo mi (znak za mikro). Definiše se umesto slova A (adresa 57352).

@

Namena: Definisanje izgleda novog karaktera.
Opšti oblik: @..niz..

Argumenti: niz — niz karaktera koji se sastoji samo od karaktera tačke i karaktera slova B.

Napomena: Ova naredba se koristi, a i objašnjena je, i u definisanju sprajtova.

Nizom se definiše osam tačaka jednog od osam horizontalnih redova karaktera. Slovo B će dati tačku karaktera boje zapisa (vidljivu tačku), a tačka u nizu će dati tačku boje pozadine (nevidljivu tačku).

Primer: Videti primer za naredbu **DESIGN 2.**

6.2.14 **Sprajtovi**

Stvaranje sprajtova (DESIGN 0, DESIGN 1, @, MOB SET, MOB OFF, CMOB)

DESIGN 0, DESIGN 1

Namena: Određivanje dela memorije za smeštanje slike sprajta.
Opšti oblik: **DESIGN** 0,a
DESIGN 1,a

Argumenti: 0 — za visoko rezolucijski (HRG) sprajt.
1 — za višebojni (MULTI COLOUR) sprajt.
a — adresa prvog, od ukupno 63 bajta potrebnih za smeštanje sprajta.

Sprajtovi su specijalna vrsta novo definisanih karaktera. Za razliku od običnih karaktera koji su veličine 8 puta 8 tačaka, sprajtovi su veličine:

24 (po horizontali) puta 21 (po vertikali) za HRG.

12 (po horizontali) puta 21 (po vertikali) za MULTI COLOUR.

Za definisanje jednog sprajta sledi da je potrebno 24 × 21 bit, odnosno 24 × 21/8 bajtova, tj. 63 bajta. Za smeštanje sprajtova koriste se delovi memorije (koji se u radu sa sprajtovima nazivaju blokovi) od 64 bajta.

Svaki karakter se sastoji od 8 puta 8 tačaka koje mogu biti vidljive ili nevidljive. Tačka je vidljiva tj. ispisana u boji zapisa ako je njoj odgovarajući bit u memoriji postavljen na jedinicu, odnosno nije vidljiva (boje je osnove) ako je bit jednak nuli. Za 64 tačke tj. bita karaktera potrebno je 8 bajtova. Velikom Komodorovom slovu A odgovaraju sledeći bajtovi i njima odgovarajuće binarne, decimalne i heksadecimalne vrednosti.

adresa	sadržaj	dec.	hex.
53256	00011000	24	$18
53257	00111100	60	$3C
53258	01100110	102	$66
53259	01111110	126	$7E
53260	01100110	102	$66
53261	01100110	102	$66
53262	01100110	102	$66
53263	00000000	0	$00

Uočava se da binarne vrednosti jedan definišu slovo A.

Napomena: karakter ROM se nalazi iza adresnog područja kola VIC, SID, CIA i iza kolor RAM-a (videti organizaciju memorije) pa se ne može direktno očitavati.

Primer:

```
10 A=57344+8
20 POKE A+0,%00000000
30 POKE A+1,%00111100
40 POKE A+2,%01000010
50 POKE A+3,%01000010
60 POKE A+4,%01111110
70 POKE A+5,%01000010
80 POKE A+6,%01000010
90 POKE A+7,%00000000
```

U datom primeru definiše se novi karakter slova A. Osam odgovarajućih jednobajtnih vrednosti upisuje se od adrese (57352) od koje se nalazi prekopiran Komodorov karakter slova A. Po upisivanju programa prvo je potrebno direktno izvršiti naredbu MEM, a zatim startovati program. Time će sva slova A postati novo definisana slova A. Za vraćanje u normalni način rada potrebno je izvršiti naredbu **NRM.** Nakon toga dati primer može prikazati kako se dobija novo definisano A na visoko rezolucijskom ekranu. Potrebno je dodati sledeće dve linije:

```
5 HIRES 0,1
100 GOTO 100
```

DESIGN 2

Namena: Definisanje novog karaktera.
Opšti oblik: DESIGN 2, a

Argumenti: a — adresa karaktera, preslikanog u RAM naredbom **MEM,** koji se želi promeniti.

Napomena: Naredba **DESIGN** se koristi, a i objašnjena je, i u definisanju sprajtova **(DESIGN 0 i DESIGN 1).**

Preslikani karakteri nalaze se od adrese 57344 ($E000). Svakom karakteru odgovara po 8 bajtova. Karakteri su poređani po redu koji je dat u tabeli ekranskih kodova i njima odgovarajućih karaktera (u dodatku), s tim da su prvo karakteri prvog, a zatim drugog seta. Adresa prvog karaktera može se naći po izrazima:

Opšti oblik: TEXT x, y, „string", n, a, x1

Argumenti: x — x koordinata leve ivice prvog karaktera.
y — y koordinata gornje ivice prvog karaktera.
string — niz karaktera koji se ispisuje.
n — način crtanja.
a — povećanje karaktera po vertikali. Za vrednosti 0 i 1 ispisuju se bez povećanja.
x1 — rastojanja između levih ivica karaktera.
Sve ostalo navedeno za argumente u naredbi **PLOT** važi i u ovoj naredbi.

Napomene: Za prelazak na drugi set karaktera (mala slova), treba u string uneti **CTRL A** (na ekranu se ispisuje inverzno A). Za vraćanje na osnovni set karaktera treba uneti **CTRL B** (na ekranu se ispisuje inverzno B).

Primer:
```
10 HIRES 0,1
20 FOR I=6 TO 1 STEP -1
30 TEXT 6*I-5,25,"ABCDDEFG",1,I,40
40 NEXT
50 GOTO 50
```

6.2.13 Definisanje novih karaktera

Stvaranje novih karaktera za tekstualni način rada (MEM, DESIGN 2, @)

MEM

Namena: Preslikavanje oba karakter seta iz ROM memorije u RAM memoriju.
Opšti oblik: MEM

Napomena: Poništavanje dejstva ove naredbe ostvaruje se naredbama **NRM, CSET 0** ili **CSET 1.**

Prebacivanjem karaktera iz memorije čiji se sadržaj ne može menjati (ROM) u memoriju čiji se sadržaj može menjati (RAM) omogućuje se promena karaktera tj. kreiranje novih karakter setova.

Izvršenjem naredbe MEM dešava se sledeće:

— Karakteri iz ROM memorije preslikavaju se u RAM memoriju od adrese 57344 ($E000). To je memorija iza Kernal ROM-a, koja se već koristi za visoko rezolucijski ekran.

— Video memorija (tekstualni ekran) postavlja se od 52224 ($CC00) do 53247 uz prethodno brisanje tog dela memorije. Potrebno je obratiti pažnju da eventualno ne dođe do preklapanja sa nekim mašinskim programom koji je tu prethodno smešten (za ostvarenje do sada navedenog, kontrolni registar VIC kola na adresi 53272, postavljen je na vrednost %00111000).

— Sprajtovi se smeštaju od adrese 49152 ($C000). Za dodatna objašnjenja videti naredbe za rad sa sprajtovima.

Izvršenjem naredbe karakteri su preslikani u memoriju koju koristi visoko rezolucijski, tj. višebojni ekran. To se može proveriti direktnim zadavanjem naredbi:

MEM:CSET 2:PAUSE 100

Iz tog razloga nije moguće upotrebljavati novo definisane karaktere pri radu u visokoj rezoluciji. Izvršenjem naredbe **HIRES** navedeni deo memorije se briše, a time i preslikani karakteri.

Komodor omogućava definisanje i istovremeno postojanje 256 sprajta, s tim što se na ekranu iz Sajmons bejzika može istovremeno prikazati do 8 sprajtova (mašinskim programiranjem taj broj se može povećati). Tih 256 sprajtova, a i njima odgovarajući blokovi memorije označeni su od 0 do 255. Ograničenja koja unosi konstrukcija računara i organizacija memorije dozvoljavaju korišćenje samo sledećih blokova, tj. sprajtova:

– u tekstualnom načinu rada:

blokovi	adrese
13 do 15	832 do 1023 ($0340 do $03FF)
64 do 255	4096 do 16383 ($1000 do $3FFF)

– u grafičkom načinu rada (HRG i MULTI COLOUR):

blokovi	adrese
16 do 63	50176 do 53247 ($C400 do $CFFF)

– u tekstualnom radu sa novo definisanim karakterima:

blokovi	adrese
16 do 47	50176 do 52223 ($C400 do $CBFF)
192 do 255	61440 do 65535 ($F000 do $FFFF)

Pri korišćenju pojedinih blokova treba imati u vidu da ne dođe do preklapanja sa potrebnim podacima u memoriji. To je od posebnog značaja za blokove 64 do 255 u tekstualnom načinu rada, da ne bi došlo do preklapanja sa bejzik programom (videti organizaciju memorije)

Konačno, na osnovu izloženog, vrednost argumenta a (adresu) moguće je dobiti po izrazu:

$$a = ba + 64*b$$

gde je ba bazna adresa video memorije i iznosi:

ba=0	za tekstualni način rada
ba=49152 ($C000)	za grafički način rada i rad sa definisanim karakterima.

Primer:

```
100 DESIGN 0,64*13
110 @........................
120 @........................
130 @........................
140 @....BBBB.......BBBB.....
150 @....B...BB...BB...BB....
160 @....B.....B.B.....BB....
170 @....B......B......BB....
180 @....B.............BB....
190 @....B.............BB....
200 @....B....B..B.....BB....
210 @....B....BRRBB....BB....
220 @....B...BBBBBBB...BB....
230 @....B...BB.B.BBB..BB....
240 @....B..BBB....B...BB....
250 @....B.BBB.........BB....
260 @....BBBBB......BBBBB....
270 @....BBBBBB...BBBBBBB....
280 @....BBB..BB.B...........
290 @...BBB.....B............
300 @..BB....................
310 @........................
320 :
400 COLOUR 7,7
410 MOB SET 0,13,5,0,0
420 MMOB 0,0,0,300,205,0,100
```

U datom primeru formira se HRG sprajt (znak Mikro knjige). Sprajt je svetlo plave boje, dodeljen mu je blok memorije 13, a označen je brojem 0. Sprajt će se prikazivati na tekstualnom ekranu.

@

Namena: Definisanje izgleda sprajta.
Opšti oblik: @..niz...

Argumenti: niz — niz karaktera koji se sastoji od karaktera tačke i karaktera slova B, C, i D.

Napomena: Ova naredba se koristi, a i objašnjena je i u definisanju karaktera.

Nizom se definiše 24 tačaka (po horizontali) HRG sprajta, odnosno 12 tačaka (takođe po horizontali) višebojnog (MULTI COLOUR) sprajta. Pri tome niz se sastoji od 24, odnosno 12 karaktera. Karakteri u nizu određuju boje tačaka sprajta na sledeći način:

HRG sprajt (dve boje)

karakter	
.	tačka boje pozadine
B	postavlja se naredbom **MOB SET**

MULTI COLOUR sprajt (četiri boje)

karakter	
.	tačka boje pozadine
B	postavlja se naredbom **CMOB**
C	postavlja se naredbom **MOB SET**
D	postavlja se naredbom **CMOB**

Potrebno je upotrebiti ovu naredbu 21 put za definisanje svakog od 21 redova od kojih se sastoji sprajt.

Primer:

```
100 DESIGN 1,64*13
110 @B.....CCCCCD
120 @B.....CCCCCD
130 @BB....CCCCDD
140 @BB....CCCCDD
150 @BBB...CCCDDD
160 @BBB...CCCDDD
170 @BBBB..CCDDDD
180 @BBBB..CCDDDD
190 @BBBBB.CDDDDD
200 @BBBBB.CDDDDD
210 @BBBBBBDDDDDD
220 @DDDDDDBBBBBB
230 @DDDDDC.BBBBB
240 @DDDDDC.BBBBB
250 @DDDDCC..BBBB
260 @DDDDCC..BBBB
270 @DDDCCC...BBB
280 @DDDCCC...BBB
290 @DDCCCC....BB
300 @DDCCCC....BB
310 @DCCCCC.....B
320 :
400 MOB SET 1,13,0,1,1
410 CMOB 5,7
420 MMOB 1,0,0,300,200,0,100
```

U navedenom primeru formira se višebojni sprajt. Smešten je u trinaesti blok, što znači da će se prikazivati na tekstualnom ekranu, a označen je brojem 1. Naredbom **MOB**

SET zadata je crna boja, a naredbom **CMOB** zelena i žuta boja. Sprajt takođe poseduje i boju pozadine.

MOB SET

Namena: Uključivanje (prikazivanje) i zadavanje osobina sprajta.

Opšti oblik: **MOB SET** s,b,c,p,n

Argumenti: s – broj od 0 do 7 kojim se označava sprajt.
b – broj bloka memorije za smeštanje sprajta (videti naredbu **DESIGN 0, DESIGN 1**).
c – broj od 0 do 15 koji određuje boju sprajta. (videti naredbu za definisanje izgleda sprajta @).
p – prioritet u iscrtavanju sprajta. Za p=0 sprajt je zaklonjen likom na ekranu, a za p=1 sprajt zaklanja lik na ekranu.
n – vrsta sprajta. Za n=0 sprajt je u dve boje (HRG), a za n=1 sprajt je četvorobojan (MULTI COLOUR).

Primer: Videti primere za naredbe **DESIGN 0, DESIGN 1** i @.

MOB OFF

Namena: Prestanak prikazivanja sprajta.

Opšti oblik: **MOB OFF**, s

Argumenti: s – broj od 0 do 7 koji označava sprajt (broj sprajta).

Primer: Po izvršenju primera u naredbi @ izvršiti naredbu **MOB OFF 1.**

CMOB

Namena: Zadavanje dodatnih boja višebojnom (MULTI COLOUR) sprajtu.

Opšti oblik: **CMOB** c1,c2

Argumenti: c1 – broj od 0 do 15 koji određuje boju tačaka sprajta zadatih slovom B u naredbi @.
c2 – broj od 0 do 15 koji određuje boju tačaka sprajta zadatih slovom D u naredbi @.

Napomene: Ove boje su iste za sve višebojne sprajtove.

Primer: Videti primer u naredbi za definisanje izgleda sprajta, @.

Upotreba sprajtova (MMOB, RLOCMOB, DETECT, CHECK)

MMOB

Namena: Prikazivanje i pomeranje sprajta na ekranu.

Opšti oblik: **MMOB** s,x,y,x1,y1,p,v

Argumenti: s – broj sprajta (od 0 do 7).
x – početna x koordinata gornjeg levog ugla sprajta.
y – početna y koordinata gornjeg levog ulga sprajta
x1 – krajnja x koordinata gornjeg levog ugla sprajta
y1 – krajnja y koordinata gornjeg levog ugla sprajta.
p – povećanje sprajta.
p=0 bez povećanja.

p=1 povećanje u horizontalnom pravcu 2 puta.
p=2 povećanje u vertikalnom pravcu 2 puta.
p=3 povećanje u oba pravca 2 puta.

v — broj od 0 do 255 koji određuje brzinu pomeranja sprajta na ekranu. Za v=0 brzina je najveća a, za v=255 najmanja.

Napomene: Koordinatni sistem u kome se određuju položaji sprajtova nije isti kao HRG, tj. MULTI COLOUR sistem. Koordinatne ose su orijentisane u istom pravcu, ali koordinatnom početku HRG sistema odgovara tačka sa koordinatama x=24 i y=50 u koordinatnom sistemu sprajtova. Sa druge strane y koordinata je ograničena na 255, x koordinata je praktično neograničena. Sve to dozvoljava postojanje sprajtova i van ekrana. Za detaljnije informacije o koordinatnom sistemu sprajtova videti poglavlje o grafici.
Primer: Videti primere u naredbama **DESIGN 0, DESIGN 1** i **@**.

RLOCMOB

Namena: Pomeranje sprajta na ekranu.
Opšti oblik: RLOCMOB s,x,y,p,v

Argumenti: s — broj sprajta (od 0 do 7).
x — x koordinata na koju se pomera sprajt (njegov gornji levi ugao).
y — y koordinata na koju se pomera sprajt.
p — povećanje sprajta (identično kao u naredbi **MMOB).**
v — brzina pomeranja sprajta (identično kao u naredbi **MMOB).**

Napomene: Identično kao u naredbi **MMOB.**

Primer: Po izvršenju primera za naredbu **@** dopisati sledeće linije i izvršiti ih naredbom **GOTO 500.**

```
500 RLOCMOB 1,100,100,0,10
510 RLOCMOB 1,200,200,0,20
520 GOTO 500
```

DETECT

Namena: Zadavanje tipa sudara koji će se ispitivati.
Opšti oblik: DETECT k

Argumenti: k — broj kojim se zadaje da li treba ispitati sudar sprajta sa sprajtom ili sudar sprajta sa likom na ekranu.
k=0 sudar sprajta sa sprajtom.
k=1 sudar sprajta sa likom na ekranu.

Napomene: Naredbu treba zadati programski, pre naredbe za ispitivanje sudara **CHECK.**

Primer:

```
10 PROC SPRAJT
20 PRINT CHR$(147)
30 COLOUR 3,3
40 PRINT AT(10+15*RND(0),9+5*RND(0))"O"
100 DESIGN 1,64*13
110 @BBBB........
120 @BBBBBB......
130 @BBBBBBBB....
140 @BBBBBBBBBBBB
```

```
150 @BBBBBBBB....
160 @BBBBBB......
170 @BBBB........
180 @CCCC........
190 @CCCCCC......
200 @CCCCCCCC....
210 @CCCCCCCCCCCC
220 @CCCCCCCC....
230 @CCCCCC......
240 @CCCC........
250 @0000........
260 @000000......
270 @00000000....
280 @000000000000
290 @00000000....
300 @000000......
310 @0000........
320 :
330 MOB SET 1,13,0,1,1
340 CMOB 5,7
350 MMOB 1,30,130,30,130,0,0
360 FOR I=30 TO 350
370 : RLOCMOB 1,I,130,0,0
380 : DETECT 1
390 : IF CHECK(1)=0 THEN CALL SUDAR
400 NEXT
410 CALL SPRAJT
420 PROC SUDAR
430 : FOR I=1 TO 15
440 :   COLOUR I,I-1
450 :NEXT
460 CALL SPRAJT
```

CHECK

Namena: Ispitivanje sudara između sprajta i sprajta ili sprajta i pozadine.

Opšti oblik: **CHECK** (s1,s2)
CHECK (s)

Argumenti: s1,s2 — brojevi sprajtova (od 0 do 7) za koje se ispituje da li su se sudarili.
s — broj sprajta (od 0 do 7) koji se ispituje da li se sudario sa pozadinom.

Napomene: Između reči **CHECK** i leve zagrade ne sme postojati prazno polje.

Ovo je funkcijska naredba koja daje rezultat 1, ako nije došlo do sudara, a rezultat 0 ako je došlo do sudara. U dodiru sprajta sa pozadinom (sa pozadinom i nacrtanim likovima) ne dolazi do sudara sa tačkama pozadine 00 i 01 (videti naredbu **MULTI**). Time je omogućeno formiranje likova sa kojima neće dolaziti do sudara. Ako je za argumente s1 i s2 naveden isti broj sprajta ispitivaće se sudar tog sprajta sa bilo kojim sprajtom.

Primer: Videti primer za naredbu **DETECT.**

6.2.15 Zvuk

Naredbe za dobijanje zvukova (VOL, WAVE, ENVELOPE, MUSIC, PLAY)

VOL

Namena: Podešavanje glasnosti zvuka.

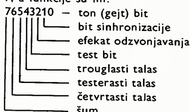
Opšti oblik: **VOL** n

Argumenti: n — ceo broj između 0 i 15. Nula isključuje zvuk, a 15 daje najglasniji zvuk.
Napomena: Signali iz sva tri generatora podešavaju se na istu amplitudu.
Primer: Videti primer za naredbu **PLAY.**

WAVE

Namena: Zadavanje talasnog oblika zvuka.
Opšti oblik: WAVE z,b
Argumenti: z — broj koji određuje upotrebljeni izvor zvuka (generator zvuka).
z=1 — generator 1.
z=2 — generator 2.
z=3 — generator 3.

b — binarni broj sa osam cifara. Svaka cifra (0 ili 1) ima posebnu namenu u zadavanju karaktera zvuka. Cifre su označene brojevima od 0 do 7, a funkcije su im:



bit 0 — Ovim bitom određuje se talasni oblik signala. Ako je postavljen na jedinicu jačina zvuka raste do maksimalnog nivoa (attack), opada do konstantnog nivoa (decay) i održava konstantni nivo (sustain). Ako je bit jednak nuli, jačina zvuka će otpočeti opadanje od konstantnog nivoa do nule (release).

bit 1 — Ovim bitom ostvaruje se sinhronizacija između zvukova različitih zvučnih izvora i to na sledeći način:
za zvuk: 1 — sinhronizuje 1 sa 3
2 — sinhronizuje 2 sa 1
3 — sinhronizuje 3 sa 2

bit 2 — Postavljanjem ovog bita na jedinicu postiže se efekat odzvonjavanja (ring modulacija) između pojedinih zvukova. Za dalje detalje videti poglavlje 10.

bit 3 — Postavljanjem test bita na jedinicu isključuje se generisanje zvuka. Ovaj bit se obično ne koristi iz Sajmons bejzika.

bit 4 — Postavljanjem ovoga bita na jedinicu dobija se signal trouglastog oblika.

bit 5 — Postavljanjem ovoga bita na jedinicu dobija se signal testerastog talasnog oblika.

bit 6 — Postavljanjem ovoga bita na jedinicu dobija se signal četvrtastog talasnog oblika.

bit 7 — Postavljanjem ovoga bita na jedinicu dobija se beli šum.

Primer: Videti primer za naredbu **PLAY.**

ENVELOPE

Namena: Zadavanje oblika obvojnice zvuka.
Opšti oblik: ENVELOPE z,a,d,s,r

Argumenti: z — broj koji određuje upotrebljeni izvor zvuka (1, 2 ili 3).
a — broj od 0 do 15, koji zadaje trajanje porasta jačine zvuka do maksimalne jačine (attack).
d — broj od 0 do 15, koji zadaje trajanje opadanja jačine zvuka do konstantnog nivoa (decay).
s — broj od 0 do 15, koji zadaje trajanje konstantne jačine zvuka (sustain).
r — broj od 0 do 15, koji zadaje trajanje opadanja jačine zvuka do nestanka zvuka (release).

Primer: Videti primer za naredbu **PLAY.**

MUSIC

Namena: Zadavanje nota i trajanja njihovog izvođenja.
Opšti oblik: MUSIC n, „string"

Argumenti: n — broj koji određuje trajanje pojedine note. Trajanje note u sekundama može se izračunati delenjem ovog broja sa 12.

string — string kojim se određuju note tj. tonovi generisanog zvuka. Prvi karakter mora biti kontrolni karakter za brisanje sadržaja ekrana:
CLR (SHIFT CLR/HOME).
Drugi karakter je broj upotrebljenog zvučnog izvora i može biti:
1, 2 ili 3.

Nakon toga navode se grupe od po tri karaktera, pri čemu svaka grupa definiše po jednu notu tj. ton. Prvi karakter u grupi određuje visinu tona u oktavi, i može biti:

C,D,E,F,G,A ili H.

Ukoliko se unese uz pritisnut taster SHIFT ton će biti povišen za pola. Drugi karakter u grupi je cifra koja određuje jednu od osam mogućih oktava i može biti:
0,1,2,3,4,5,6 ili 7.
Treći karakter u grupi određuje trajanje tona, i zadaje se preko funkcijskih tastera. Pri tome važi:

taster	trajanje	na ekranu
f1	1/16	inverzno E
f3	1/8	inverzno F
f5	1/4	inverzno G
f7	1/2	inverzno H
f2	1/1	inverzno I
f4	2/1	inverzno J
f6	4/1	inverzno K
f8	8/1	inverzno L

Trajanje se dobija u delovima ili umnošcima osnovnog trajanja note. Dobijeni inverzno ispisani karakteri nalaze se u drugom setu karaktera, što znači

da će se na ekranu dobiti kao takvi ako su pritisnuti tasteri za prelazak na drugi set karaktera (zajedno C= i SHIFT). Kao poslednji karakteri u stringu za zadavanje nota može se navesti:
SHIFT CLR/HOME i G
što će startovati **RELEASE** period zadnje note.

String za zadavanje nota može biti dat ne samo direktno već i u obliku string promenljive ili izraza sa stringovima.

Primer: Videti primer za naredbu PLAY.

PLAY

Namena: Izvršavanje (sviranje) zadatih nota.
Opšti oblik: PLAY n

Argumenti: n — broj koji određuje načine sviranja i ima sledeće vrednosti i namene:

n=1 Izvršavaju se zadate note. Po završenom sviranju prelazi se na dalje izvršavanje programa.

n=2 Izvršavaju se zadate note ali i bejzik program (prekidnim načinom rada ostvareno je paralelno izvršavanje nota i bejzik programa). Pri tome ako se ne izvršava bejzik program neće se izvršavati ni zadate note.

Ovom naredbom se izvršavaju note zadate naredbom **MUSIC.** Naredba se istovremeno odnosi na sva tri zvučna izvora.

Primer:

```
100 VOL 15:REM JACINA ZVUKA
110 :
120 REM PAZNJA, ZNAK a OZNACAVA PRITISNUTO SHIFT A
130 A$-"(CLR)1C3(F5)F3(F7)G3(F5)a3(F5)A3(F5)A3(F3)G3(F3)F3(F5)G3(F5)A3(F7)a3"
140 A$-A$+"(F5)O4(F5)C4(F5)C4(F3)a3(F3)A3(F5)C4(F5)C4(F5)a3(F3)A3(F3)a3(F5)"
150 A$-A$+"C4(F5)A3(F7)BG(F7)"
160 B$-"C4(F5)A3(F7)C4(F5)E4(F5)O4(F5)D4(F3)C4(F3)a3(F5)a3(F5)G3(F7)a3(F5)O4"
170 B$-B$+"(F5)F4(F5)C4(F3)a3(F3)G3(F5)C4(F5)C4(F5)a3(F3)G3(F3)a3(F5)C4(F5)"
180 B$-B$+"G3(F7)(F5)3G(F1)"
190 :
200 REM TALASNI OBLIK
210 WAVE 1,00100001
220 WAVE 2,01000011
230 ENVELOPE 1,2,5,5,9
240 ENVELOPE 2,0,5,5,9
250 :
260 MUSIC 6,A$+B$+B$
270 PLAY 2
280 PRINT CHR$(147)
290 REPEAT
300 :  PRINT A;
310 :  A-A+1
320 UNTIL PEEK (198)-1
330 VOL 0
340 ENO
```

6.2.16 Rad sa diskom i kasetofonom

Olakšavanje rada sa diskom (DISK, DIR)

DISK

Namena: Olakšavanje rada sa disk jedinicom.
Opšti oblik: DISK „naredba"

Argumenti: naredba – disk komanda

Uobičajeni način komunikacije sa diskom zahteva zadavanje nešto dužeg niza naredbi. Na primer, za formatiranje diskete potrebno je izvršiti sledeće naredbe:

OPEN 1,8,15, "N:PRVIDISK,01"
CLOSE 1

Upotrebom naredbe **DISK** postupak se pojednostavljuje. To je prikazano na sledećim komandama diska:

FORMATIRANJE DISKETE (new)
Opšti oblik: DISK „N:ime,id"

Argumenti: ime – željeno ime diskete (do 16 karaktera).
id – željena oznaka diskete (do 2 karaktera).

Primer: DISK „N:SIMON'S DISKETA, 10A"

BRISANJE DATOTEKE (scratch)
Opšti oblik: DISK „S:ime 1,ime 2,...."

Argumenti: ime 1 – ime datoteke koja se briše.
ime 2 – ime datoteke koja se briše.

Primer: DISK „S:PROBNIFAJL"

DODELJIVANJE NOVOG IMENA DATOTECI (rename)
Opšti oblik: DISK „R:novo ime=staro ime"

Argumenti: novo ime – novo ime koje se dodeljuje.
staro ime – ime koje se briše, prethodno ime.

Primer: DISK „R:HEMIJA 2.2=HEMIJA 2.1"

INICIJALIZACIJA DISKETE (initiate)
Opšti oblik: DISK „I"

REORGANIZACIJA DISKETE (validate)
Opšti oblik: DISK „V"

DIR

Namena: Prikazivanje sadržaja diskete, bez brisanja programa u memoriji.
Opšti oblik: DIR „string"

Argumenti: string – string može biti sledeći i ima značenja:
$ – prikazuje se kompletan sadržaj diskete.

$:ime – potvrđuje se da li je na disketi datoteka navedenog imena. Znak pitanja (?) može zameniti bilo koji karakter u imenu. Može se upotrebiti i zvezdica (*), s tim što će ona zameniti sve karaktere do kraja imena.

S:ime=tip — potvrđuje se da li je na disketi datoteka navedenog imena i navedenog tipa. Tipovi mogu biti: P (programska), S (sekvencijalna), R (relativna) i U (korisnička). Znaci za zamenu karaktera (?) i za zamenu dela imena (*) mogu biti upotrebljeni.

Ovom naredbom prikazuje se katalog (direktorijum) diska. Zamenjuje standardnu bejzik naredbu **LOAD** „S", 8. **LIST**

Primer: DIR „S"

Datim primerom dobija se spisak svih datoteka na disku.

Snimanje sadržaja ekrana (SCRSV, SCRLD)

SCRSV

Namena: Snimanje nisko rezolucijskog sadržaja ekrana na spoljnjem uređaju (kasetofon ili disk jedinica).
Opšti oblik: SCRSV d,p,a, „ime,S,W"

Argumenti: d — logički broj datoteke (od 1 do 127).
p — broj spoljnog (perifernog) uređaja. Za kasetofon je 1, a za disk jedinicu 8.
a — sekundarna adresa. Za kasetofon je 1, a za disk jedinicu 2.
ime — ime pod kojim se snima sadržaj ekrana.

Ovom naredbom snima se samo nisko rezolucijski sadržaj ekrana (tekstualni način rada ekrana). Visoko rezolucijski sadržaj (HRG) i višebojni (MULTI COLOR) sadržaj ne mogu se snimiti ovom naredbom. Učitavanje slike snimljene ovom naredbom obavlja se naredbom **SCRLD.**

Primer:

```
10 PRINT CHR$(147)
20 FILL 5,10,10,15,1,7
30 SCRSV 1,8,2,"PROBA,S,W"
40 GOTO 40
```

SCRLD

Namena: Učitavanje sadržaja nisko rezolucijskog ekrana snimljenog naredbom **SCRSV.**
Opšti oblik: SCRLD d,p,a, „ime"

Argumenti: d — logički broj datoteke (od 1 do 127).
p — broj spoljnog (perifernog) uređaja. Za kasetofon je 1, a za disk jedinicu 8
a — sekundarna adresa. Za kasetofon je 1, a za disk jedinicu 2.
ime — ime pod kojim je snimljen sadržaj ekrana.

Primer:

```
10 SCRLD 1,8,2,"PROBA
20 GOTO 20
```

6.2.17 **Rad sa štampačem**

Ispisivanje teksta sa ekrana i kopiranje ekrana (HRDCPY, COPY)

HRDCPY

Namena: Štampanje nisko rezolucijskog sadržaja ekrana na štampaču.
Opšti oblik: HRDCPY

Ovom naredbom štampaju se karakteri sa ekrana na papir odgovarajućeg štampača (GEMINI 10C, CBM VC-1526, MPS 801, SEIKOSHA GP100 VC).

COPY

Namena: Ovom naredbom prenosi se višebojni (MULTI COLOUR) ili visoko rezolucijski sadržaj ekrana (HRG) na papir odgovarajućeg štampača.
Opšti oblik: COPY

Neki od štampača koji mogu biti upotrebljeni su: GEMINI 10C, CBM VC-1525, CBM VC-1520, SEIKOSHA GP100 VC, a takođe i GEMINI 10/15, EPSON RX-80 i EPSON FX-80 sa odgovarajućim interfejsom.

6.2.18 Rad sa upravljačkim uređajima

Rad sa svetlosnom olovkom i upravljačkom palicom (PENX, PENY, POT, JOY)

PENX

Namena: Određivanje x koordinate položaja svetlosne olovke.
Opšti oblik: PENX

Ovo je funkcijska naredba koja daje položaj svetlosne olovke po horizontali ekrana. Vrednost nula dobija se za krajnji levi položaj olovke na ekranu.

Za dobijanje vrednosti koordinata koje važe pri radu u visokoj rezoluciji (HRG) treba upotrebiti izraz:

$$x = (xp - 40)*2$$

gde je xp koordinata dobijena naredbom **PENX.**

Primer:

```
10 HIRES 1,2
20 PROC LINIJA
30 : X=(PENX-40)*2:Y=PENY-40
40 : IF X<0 OR X>319 OR  Y<0 OR Y>199 THEN CALL LINIJA
50 : LINE X0,Y0,X,Y,1
60 : X0=X:Y0=Y
70 : CALL LINIJA
```

U ovom programu ostvaruje se crtanje prave linije između prethodnog i poslednjeg položaja svetlosne olovke na ekranu.

PENY

Namena: Određivanje y koordinate položaja svetlosne olovke.
Opšti oblik: PENY

Ovo je funkcijska naredba koja daje položaj svetlosne olovke po vertikali ekrana. Vrednost nula dobija se za krajnji gornji položaj olovke na ekranu (na okviru).

Za dobijanje vrednosti koordinata koja važe pri radu u visokoj rezoluciji (HRG) treba upotrebiti izraz:

$$y = yp - 40$$

gde je yp koordinata dobijena naredbom **PENY.**

Primer: Videti primer za naredbu **PENX.**

POT

Namena: Određivanje položaja priključenog potenciometra.
Opšti oblik: POT (n)

Argumenti: n — broj koji odgovara priključenom potenciometru (0 ili 1).

Ovo je funkcijska naredba koja daje položaj priključenih potenciometara. Rezultuje brojnom vrednošću između 0 i 255. Granične vrednosti odgovaraju krajnjim položajima potenciometra.

Primer:
```
10 A=POT(0)
20 PRINT AT(5,5)A:GOTO 10
```

JOY

Namena: Određivanje položaja priključene upravljačke palice.
Opšti oblik: JOY

Napomene: Važi samo za palicu priključenu u upravljački ulaz 2 (control port 2).
Ovo je funkcijska naredba koja daje brojne vrednosti u zavisnosti od položaja upravljačke palice. Položaji palice i odgovarajuće vrednosti su:

položaj palice	vrednosti
gore	1
gore desno	2
desno	3
dole desno	4
dole	5
dole levo	6
levo	7
gore levo	8
miran položaj	0

Ako je pritisnut taster na palici vrednosti će biti uvećane za 128.

Primer:
```
10 IF JOY=1 THEN PRINT AT(5,5)"IDEMO GORE"
20 IF JOY=5 THEN PRINT AT(5,5)"IDEMO DOLE"
30 IF JOY=7 THEN PRINT AT(5,5)"IDEMO LEVO"
40 IF JOY=3 THEN PRINT AT(5,5)"IDEMO DESNO"
50 IF JOY>=128 THEN PRINT "TO! PALJBA"
60 GOTO 10
```

6.3 IZVEŠTAJI

Izveštaji u Sajmons bejziku, isto kao i izveštaji u standardnom Komodorovom bejziku, prijavljuju pojavu neke greške. Sajmons bejzik donosi 11 novih izveštaja koji proizilaze iz novih naredbi Sajmons bejzika.

BAD CHAR FOR A MOB (pogrešno definisan karakter ili sprajt)
Grešku treba tražiti u linijama sa naredbom za definisanje karaktera tj. sprajta (@).

BAD MODE (pogrešan način rada)
Ovo je univerzalni izveštaj u Sajmons bejziku. Odgovara izveštaju ILLEGAL QUANTITY ERROR.

END LOOP WITHOUT LOOP (END LOOP bez LOOP)
U toku izvršavanja programa naišlo se na naredbu **END LOOP**, a da prethodno nije izvršena naredba **LOOP**. Ovaj izveštaj odgovara izveštaju standardnog bejzika NEXT WITHOUT FOR.

END PROC WITHOUT EXEC (END PROC bez EXEC)
U toku izvršavanja programa naišlo se na naredbu END PROC, a da prethodno nije upotrebljena naredba EXEC. Ovaj izveštaj odgovara izveštaju standardnog bejzika RETURN WITHOUT GOSUB.

INSERT TOO LARGE (predugačko umetanje)
Ovaj izveštaj će se pojaviti u slučaju neodgovarajuće upotrebe naredbi **INST** i **INSERT**.

NOT BINARY CHAR (nisu binarni karakteri)
Ovim izveštajem prijavljuje se da nije naveden tačan broj karaktera iza naredbe %, ili da nisu navedeni odgovarajući karakteri. Još se može pojaviti pri upotrebi naredbe WAVE, u kojoj se takođe koristi znak %.

NOT HEX CHAR (nisu heks. karakteri)
Ovim izveštajem prijavljuje se da nije naveden tačan broj karaktera iza naredbe $, ili da nisu navedeni odgovarajući karakteri.

PROC NOT FOUND (nije nađena procedura)
Do pojave ovoga izveštaja će doći ako su upotrebljene naredbe za izvršavanje procedure (EXEC i CALL) koja ne postoji u programu.

STACK TOO LARGE (preveliki stek)
Ovim izveštajem označava se da je upotrebljeno suviše potprograma i/ili petlji tako da nema više slobodnih mesta na steku.

STRING TOO LARGE (predugačak string)
Ovaj izveštaj će se pojaviti u slučaju neodgovarajuće upotrebe naredbi **INST** i **INSERT**.

UNTIL WITHOUT REPEAT (UNTIL bez REPEAT)
U toku izvršavanja programa naišlo se na naredbu **UNTIL**, a da prethodno nije izvršena naredba **REPEAT**. Ovaj izveštaj odgovara izveštaju standardnog bejzika NEXT WITHOUT FOR.

6.4 PRIMER PROGRAMA U SAJMONS BEJZIKU

Struktuiranim programiranjem u Sajmons bejziku napravljen je pregledan i modularan program za izvlačenje loptica u igri loto.

```
10 REM *** L O T O ***
11 TI$="000000"
12 COLOUR 4,1
13 HIRES 0,1
14 MULTI 0,7,2
15 B=4
16 C=-16
17 FOR I=1 TO 4
18 : FOR J=1 TO 10
19 :  A=A+1
20 :  EXEC LOPTICE
21 : NEXT J
22 NEXT I
```

```
23 LINE 0,140,160,140,3
24 PAINT 0,0,3
25 LOW COL 0,1,6
26 PAINT 150,150,3
27 HI COL
28 EXEC IZVLACENJE
29 LOOP
30 : TEXT 20,190,"PRITISNI 'SPACE'",2,1,8
31 : REPEAT
32 :  GET A$
33 : UNTIL A$=" "
34 : TEXT 20,190,"PRITISNI 'SPACE'",3,1,8
35 : FOR I=1 TO 7
36 :  PAINT I*16-8 ,176,I
37 : NEXT I
38 : PAINT 9*16-8,176,1
39 : EXEC IZVLACENJE
40 EXIT IF VAL(TI$)>010000
41 END LOOP
42 STOP
46 PROC LOPTICE
47 : CIRCLE J*16-8,I*32-II,8,12,2
48 : PAINT J*16-7,I*32-11,2
49 : IF A>9 ANDC=-16 THEN B=0:C=1:A=0
50 : IF A>9 AND C=1  THEN B=0:C=2:A=0
51 : IF A>9 AND C=2  THEN B=0:C=3:A=0
52 : IF A>9 AND C=3  THEN  B=0:C=4:A=0
53 : TEXT J*16-B-8,I*32-13,CHR$(A+48),I,I,1
54 : TEXT J*16-15,I*32-13,CHR$(C+48),1,1,1
55 END PROC
56 :
57 :
58 PROC IZVLACENJE
59 : FOR I=I TO 9
60 :    PROC RANDOM
61 :     S(I)=INT(40*RND(0))+1
62 :     FOR J=0 TO 1-I
63 :      IF S(1)=S(J) THEN CALL RANDOM
64 :     NEXT J
65 : NEXT I
66 : FOR J=1 TO 7
67 :  1=5.5
68 :  EXEC TEST
69 :  EXEC LOPTICE
70 : NEXT J
71 : J=9:I=5.5
72 : EXEC TEST
73 : EXEC LOPTICE
74 END PROC
75 :
76 :
77 PROC TEST
78 : IF S(J)<10 THEN A=S(J):C=-16:B=4:END PROC
79 : IF S(J)<20 THEN A=S(J)-10:C=1:B=0:END PROC
80 : IF S(J)<30 THEN A=S(J)-20:C=2:B=0:END PROC
81 : IF S(J)<40 THEN A=S(J)-30:C=3:B=0:END PROC
82 END PROC
83 END
```

7 Programiranje na mašinskom jeziku

Mala brzina izvršavanja bejzik programa i veličina memorije koju bejzik program zauzima, glavni su razlozi za programiranje na mašinskom jeziku. Navedena ograničenja dolaze do izražaja naročito u složenim i dugačkim programima, u programima za upravljanje, regulaciju, merenje, crtanje složenih crteža i obradu velikog broja podataka i rezultata.

Pisanjem programa na osnovnom jeziku računara, mašinskom jeziku, ostvaruje se povećanje brzine izvršavanja programa i smanjenje utrošene memorije za smeštanje programa u odnosu na odgovarajući bejzik program. Pisanjem programa na mašinskom jeziku može se ostvariti potpuno iskorišćenje hardverskih mogućnosti računara, a naročito mogućnosti centralne procesorske jedinice. To su razlozi zbog kojih su komercijalni programi koji se danas nalaze na tržištu, skoro uvek pisani na mašinskom jeziku.

Sa druge strane programiranje na mašinskom jeziku je teže. Pisanje programa, njegovo testiranje, nalaženje i otklanjanje grešaka zahteva dodatna znanja i iskustva. Sve to rezultuje u dužem vremenu izrade programa. Tome treba dodati obavezno vođenje složenije dokumentacije. Konačno, ako se javi potreba za modifikovanjem programa ili za njegovim prevođenjem radi primene na drugom tipu računara, to će zahtevati više rada nego u slučaju bejzik programa.

Pisanjem programa tako da se određeni njegovi delovi pišu na mašinskom jeziku, a ostali u bejziku ili nekom drugom višem programskom jeziku predstavlja vrlo dobro rešenje za efikasnu izradu programa. Tome treba dodati i mogućnost korišćenja velikog broja potprograma pisanih na mašinskom jeziku (rutina) koji se nalaze unutar Komodora, u njegovom ROM-u.

Upotreba mašinskog programiranja na Komodoru je posebno opravdana zbog toga što Komodorov bejzik ne raspolaže naredbama koje bi podržavale velike grafičke i zvučne mogućnosti Komodora.

U daljem tekstu će biti izložena materija neophodna za uspešno programiranje na mašinskom jeziku. Takođe će biti prikazano korišćenje Komodorovih ROM rutina.

7.1 OD BEJZIKA KA MAŠINSKOM PROGRAMIRANJU

Bejzik program je sačinjen od bejzik programskih linija. Programska linija se sastoji iz bejzik naredbi i podataka nad kojima se izvršavaju naredbe. Svaka linija ima svoj broj koji joj određuje položaj u programu, a time i njen redosled u izvršavanju programskih linija. Bejzik programska linija:

10 LET A=1986

ima broj programske linije 10, bejzik naredbu dodele vrednosti (**LET**) i brojni podatak 1986.

Mašinski program je po strukturi sličan bejzik programu. Mašinski program je sačinjen od mašinskih naredbi i svaka od njih je smeštena na odgovarajući način u posebnu memorijsku lokaciju. Adresa te memorijske lokacije označava položaj naredbe u programu.

Kao što je bejzik program sačinjen od bejzik programskih linija tako se za mašinski program može reći da je sačinjen od mašinskih programskih linija. Razlika je u tome što se jedna mašinska linija sastoji samo od jedne mašinske naredbe i eventualno podatka.

Komodorova centralna procesorska jedinica, mikroprocesor 6510, izvršava 151 različitu mašinsku naredbu. Naredba je sastavljena od operacije i operanda. Operacija u naredbi je aktivnost koju mikroprocesor izvršava nad operandom, tj. podatkom. Naredbe se označavaju, a i pamte u memoriji računara u obliku brojnih vrednosti. Brojne vrednosti kojim se označavaju operacije nazivaju se kodovi operacija (engl. OP code). Operacije tj. kodovi operacija su dužine od jedan bajt. To znači da se sve operacije mogu predstaviti brojevima od 0 do 255. Operandi mogu biti dužine do 2 bajta. To znači da se za označavanje podataka mogu upotrebiti brojne vrednosti od 0 do 65535.

Pisanje mašinskog programa bi se sastojalo u formiranju niza brojnih vrednosti koje bi označavale mašinske naredbe i podatke poređane po nekom smislu. Takav način pisanja programa zahteva poznavanje odgovarajuće brojne vrednosti (koda operacije) svake naredbe. Da bi se to izbeglo uveden je simbolički mašinski jezik. U njemu se umesto kodova operacija koriste slovne oznake koje opisuju dejstvo operacija tj. naredbi. Takve oznake se nazivaju mnemoničkim oznakama.

U cilju ilustracije do sada izloženog dat je uporedni prikaz bejzik programa i ekvivalentnog mašinskog programa.

bejzik program	mašinski program				
	adrese		mnemonika	kodovi	
	decimal.	heks.		decimal	heks.
10 LET A=5	49152	C000	**LDA #5**	169,5	A9,05
20 POKE 1024,A	49154	C002	**STA 1024**	141,0,4	8D,00,04
30 RETURN	49157	C005	**RTS**	96	60

Od adrese 49152 ($C000) nalazi se mašinska naredba kojom se u registar A mikroprocesora (akumulator) stavlja vrednost 5. Naredba je smeštena u dve memorijske lokacije RAM memorije. U prvoj (adresa 49152) je smešten kôd operacije (169), a u drugoj (adresa 49153) operand (5). Mnemonička oznaka naredbe je **LDA #5.**

Od adrese 49154 ($C002) nalazi se mašinska naredba koja u memorijsku lokaciju 1024 stavlja vrednost koja se nalazi u registru A. Naredba je trobajtna, smeštena je u tri memorijske lokacije. Kôd operacije (141) je smešten u prvu lokaciju (49154), a operand, vrednosti 0 i 4, je dat u naredne dve lokacije. Mnemonička oznaka ove naredbe je **STA 1024.**

Treća naredba je jednobajtna i njen kôd (96) je smešten u memorijskoj lokaciji 49157. Sa ovom naredbom se završavaju mašinski potprogrami. Njena mnemonička oznaka je **RTS.**

Datim nizom mašinskih naredbi ostvaruje se da se u memorijsku lokaciju 1024 stavlja vrednost 5. Toj memorijskoj lokaciji odgovara prvi bajt ekranske memorije (tekstualnog ekrana). Time se postiže da se u levo gornje karakter polje ekrana upisuje slovo čiji je ekranski kod 5, a to je slovo E.

Navedeni mašinski program ima za cilj da pokaže da se mašinski program sastoji iz niza osmobitnih vrednosti (bajtova) smeštenih u memorijskim lokacijama. Te vrednosti predstavljaju operacije (naredbe) i operande (podatke).

Proces sinteze mašinskog programa se sastoji u pisanju naredbi na simboličkom mašinskom jeziku. Tako napisan program se zatim prevodi u odgovarajuće brojne vrednosti – kodove. Dobijeni mašinski kodovi se zatim smeštaju u određeni deo RAM-a.

Kraći programi se mogu prevesti uz pomoć tabele kodova. To je tabela u kojoj je dat spisak svih naredbi i njima odgovarajućih kodova. Dobijeni kodovi se pomoću bejzik naredbe POKE mogu smestiti u memoriju. Unošenje dužih programa na ovakav način je dugotrajno, a mogućnosti greške su velike. Iz tog razloga smeštanje kodova u memoriju se prepušta računaru. To se postiže uz pomoć posebnih programa za unošenje brojnih vrednosti u memoriju računara. Takvi programi se uobičajeno nazivaju punjači (engl. loader). Sa njima se unose decimalne ili heksadecimalne vrednosti.

Program za punjenje ubrzava unošenje kodova, ali ne omogućava njihovo dobijanje na osnovu mnemoničkih simbola. To obavlja program koji se naziva asembler. Upotrebom asemblera program se piše na simboličkom mašinskom jeziku. Zatim se prevodi na mašinski jezik, u kodove koji se smeštaju na odgovarajuće adrese. Upotreba asemblera olakšava pisanje mašinskog programa jer on sadrži potprogram za unošenje naredbi i podataka (editor). U toku rada se signaliziraju greške ispisivanja i formiranja asemblerskog formata. Velika pogodnost koju donosi asembler je mogućnost labeliranja, tj. pridodavanja imena adresama memorijskih lokacija. Time se željena lokacija u memoriji može adresirati tj. pozivati navođenjem njene labele.

Pisanje dužih mašinskih programa uz pomoć asemblera je jedini prihvatljivi način. Nepogodnosti koje se mogu javiti kod korišćenja asemblera su: asembler zauzima nekoliko kilobajta radne memorije, potrebno ga je pre rada učitati u memoriju računara, a takođe za rad sa asemblerom potrebno je upoznati se sa asemblerskim komandama.

Još jedan koristan program pri razvijanju mašinskih programa je disasembler. On na osnovu mašinskih kodova prikazuje simbolički mašinski program (mnemoniku). Poseduje i niz drugih korisnih mogućnosti, kao što je pregled i promena sadržaja memorije i registara mikroprocesora.

Dobijeni kodovi, koji su ustvari sam mašinski program, nazivaju se i obdžekt kôd (engl. object code). Startovanje mašinskog programa se ostvaruje bejzik naredbom **SYS** a, gde je a adresa od koje se želi izvršavanje mašinskog programa. Može se upotrebiti, sa nešto izmenjenom namenom, i bejzik naredba **USR.**

7.2 BROJNI SISTEMI

Za razumevanje načina rada Komodora i programiranje na mašinskom jeziku potrebno je poznavanje binarnog i heksadecimalnog brojnog sistema, kao i načina na koji se predstavljaju brojevi. Kroz brojne sisteme objasniće se apsolutna binarna forma, forma komplementa dvojke i petobajtna forma predstavljanja brojeva. Prve dve forme su usvojene za sve osmobitne mikroračunare, a petobajtna forma je karakteristika Komodora i nekih drugih kućnih računara.

Binarni brojevi

Brojni sistem zasnovan na osnovi 10 naziva se decimalnim brojnim sistemom. Osnova deset znači da se pri brojanju posle svakih deset jedinica vrši prenos u viši težinski razred. U decimnalnom sistemu težine pojedinih težinskih razreda su 1, 10, 100,... i one su stepeni osnove deset: 10^0, 10^1, 10 ,.... Osnova deset označava takođe i da se koristi deset cifara za formiranje brojeva. U decimalnom brojnom sistemu to su cifre: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

Broj 125 se u decimalnom sistemu predstavlja:

$1 * 10^2 + 2 * 10^1 + 5 * 10^0$

Binarni sistem je zasnovan na osnovi dva. To znači da se posle dve jedinice vrši prenos u viši težinski razred. Težine pojedinih razreda su stepeni osnove dva: 2^0, 2^1, 2^2,..., a to su vrednosti: 1, 2, 4, 8,... U binarnom sistemu se koriste samo dve cifre, a to su: 0 i 1. Uočava se pogodnost korišćenja binarnog brojnog sistema za predstavljanje dva napona sa kojima rade elektronska kola digitalnih računara (0 kada nema napona i 1 kada ga ima).

Decimalni broj 125 se u binarnom sistemu predstavlja na sledeći način:

1*64 + 1*32 + 1*16 + 1*8 + 1*4 + 0*2 + 1*1 = 1111101

dec	bin.	heks.
0	0	0
1	1	1
2	10	2
3	11	3
4	100	4
5	101	5
6	110	6
7	111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F
16	10000	10
17	10001	11
18	10010	12

Tabela 7. 1 Upotrebni prikaz decimalnih, binarnih i heksadecimalnih brojeva

Apsolutna binarna forma

Sa osam cifara u binarnom brojnom sistemu mogu se predstaviti brojevi od 00000000 do 11111111, što u decimalnom brojnom sistemu čini opseg brojeva od 0 do 255. Osam binarnih cifara odgovara osmobitnom broju tj. jednom bajtu. S obzirom da su memorijske lokacije osmobitne tj. jednobajtne u njima mogu biti samo vrednosti od 0 do 255.

Fizička realizacija memorija i registara od po osam bita učinila je da se brojevi veći od 255, koji mogu nastati kao rezultat sabiranja, predstavljaju umanjeni za 256. Na primer broj 275 će u registru ili memoriji biti predstavljen kao broj 19. Negativni brojevi će biti predstavljeni kao da im je dodato 256. Broj −23 će se u memoriji nalaziti kao broj 233.

U konceptu apsolutne binarne forme brojevi su celobrojni i pozitivni. U slučaju osmobitnih brojeva to su brojevi od 0 do 255. Komodor može da radi i sa šesnestobitnim brojevima koji se dobijaju od dva osmobitna. Tada su to pozitivni celi brojevi u opsegu od 0 do 65535.

Binarni brojevi u komplementu dvojke

Koncept komplementa dvojke uveden je sa ciljem predstavljanja negativnih brojeva. Od osam bita, za označavanje predznaka upotrebljen je bit najveće težine. Taj bit se naziva bit znaka i za pozitivne brojeve je 0, a za negativne 1. Sa preostalih sedam bita u bajtu predstavljaju se 128 pozitivnih i 128 negativnih brojeva (0 se smatra pozitivnim brojem).

Dobijanje negativne vrednosti određenog broja obavlja se na sledeći način. Prvo se nađe komplement odgovarajućeg pozitivnog broja. To se postiže prevođenjem nula u jedinice i jedinica u nule. Tako dobijenom komplementu se doda vrednost jedan, i time je dobijen komplement dvojke željenog broja. Dat je primer za nalaženje broja −7 u komplementu dvojke.

```
+6    00000111
      11111000 komplement
      +      1
-7    11111001 komplement dvojke
```

Prvi bit sa leve strane je bit znaka. Uočava se da preostalih sedam bita koji odgovaraju vrednosti 7 nisu identični u slučaju pozitivne i negativne vrednosti.

Potpuno istim postupkom dobijanja negativnih vrednosti od pozitivnih, dobijaju se i pozitivne vrednosti od negativnih, što je od praktičnog značaja za programiranje na mašinskom jeziku.

Koncept komplementa dvojke se primenjuje i na šesnaestobitnim brojevima pri čemu je opseg brojeva od −32768 do 32767.

decimalni brojevi	binarni brojevi u komplementu dvojke
127	01111111
126	01111110
125	01111101
– – –	– – – – –
2	00000010
1	00000001
0	00000000
−1	11111111
−2	11111110
– – – –	– – – – –
−126	10000010
−127	10000001
−128	10000000

Tabela 7. 2. Uporedni prikaz decimalnih i binarnih brojeva u komplementu dvojke

Heksadecimalni brojevi

Heksadecimalni brojni sistem je zasnovan na osnovi broja šesnaest. To znači da će se posle šesnaest jedinica izvršiti prenos u viši težinski razred. Težine pojedinih težinskih razreda su 1, 16, 256, 4096,... One predstavljaju stepene osnove šesnaest. Šesnaest cifara heksadecimalnog brojnog sistema su: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, E, F. U tabeli 7.1 dat je uporedni prikaz decimalnih, binarnih i heksadecimalnih brojeva.

Da bi se decimalni broj 225 preveo u heksadecimalni oblik treba ga predstaviti na sledeći način:

$$14*16^1 + 1*16^0 = 225$$

što daje heksadecimalni broj E1 (decimalni broj 14 je broj E u heksadecimalnom brojnom sistemu).

Uobičajeno se koristi znak dolara ($) za označavanje heksadecimalnih brojeva. Pri tome se znak stavlja ispred broja. Time se decimalni broj 225 heksadecimalno izražava $E1.

Heksadecimalni brojevi se najčešće primenjuju za označavanje kodova operacija, podataka i adresa memorijskih lokacija.

Petobajtna forma predstavljanja brojeva

Brojne vrednosti koje se koriste pri radu u bejziku, celobrojne i realne, u memoriji računara zauzimaju pet bajtova memorije.

Pri predstavljanju celih brojeva (od −32768 do 32767) prvi bajt, po adresnoj lokaciji, je bajt veće težine (težine 256), a drugi bajt je bajt manje težine (težine 1). Treći, četvrti i peti bajt se ne koriste i njihova vrednost je nula. Na primer broj 258 je predstavljen sa sledećih pet vrednosti: 1, 2, 0, 0, 0. U slučaju negativnog broja za prvi bajt se uzima komplement, a za drugi bajt komplement dvojke vrednosti koje odgovaraju pozitivnom broju. Na primer broj −258 će biti predstavljen sa sledećih pet vrednosti: 254, 254, 0, 0, 0.

Sledeći program omogućuje prikazivanje broja u petobajtnoj formi:

```
10 INPUT A%: PRINT "BROJ:"A %
20 B=PEEK (45)+256*PEEK(46)
30 FOR I=1 TO 5
49 : PRINT I"-"PEEK(B+I+1)
50 NEXT:GOTO 10
```

Na osnovu sistemske promenljive VARTAB nalazi se programska promenljiva A% i ispisuje se pet bajtova kojima je ona predstavljena

Realni brojevi se predstavljaju drugom petobajtnom formom koja pokriva opseg brojnih vrednosti od oko 2.94E−39 do oko 1.7E38, kako pozitivnih tako i negativnih.

Realni broj A koji se predstavlja na ovaj način prevodi se u oblik ±M * 2↑E, gde je M mantisa broja, a E eksponent tog broja. Pri tome je eksponent ceo pozitivan broj, a mantisa broj koji je jednak ili veći od 0.5 i manji od 1. Eksponent se nalazi prema izrazu:

E=1+INT(LOG (ABS(A))/LOG 2),

a mantisa prema izrazu:

M=ABS(A)/2↑E

Prvi bajt petobajtne forme kojom se predstavlja realni broj A ima vrednost zbira E+128. Drugi bajt dobija celobrojnu vrednost prizvoda mantise i broja 256 ako je broj A negativan, a ako je broj A pozitivan od proizvoda se oduzima 128. Decimalni ostatak dobijen nalaženjem prethodne vrednosti množi se sa 256. Celobrojni deo je vrednost trećeg bajta, a od decimalnog ostatka se na isti način formira četvrti i peti bajt.

Zamenom promenljive A% promenljivom A, gornji program može poslužiti za prikazivanje petobajtne forme kojom se predstavljaju realni brojevi. Na ovaj način predstavljeni brojevi nazivaju se brojevi sa pokretnim zarezom (engl. floating point).

7.3 MIKROPROCESOR 6510

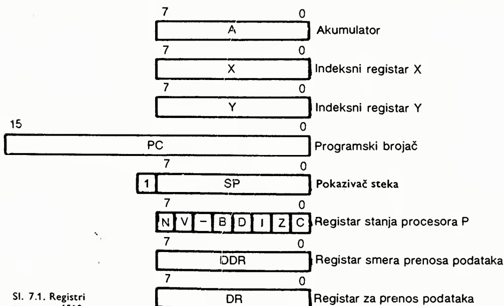
Centralna procesorska jedinica (CPU) je glavni deo računarskog sistema. Njena uloga je da na osnovu naredbi, koje se nalaze u memoriji računara, izvršava željene operacije. Te

operacije se izvršavaju nad podacima i rezultuju u transformaciji podataka ili prebacivanju podataka sa jednog mesta na drugo.

U Komodoru centralna procesorska jedinica je mikroprocesor 6510. U njemu se uočavaju tri celine: registri, aritmetičko — logička jedinica i kontrolna jedinica.

Registri

Mikroprocesor 6510 sadrži pet osmobitnih i jedan šesnaestobitni registar. Takođe poseduje i dva ulazno izlazna registra koji su sastavni deo memorije.



Sl. 7.1. Registri procesora 6510

A — Akumulator

Akumulator (engl. accumulator) je najvažniji i najčešće korišćeni registar mikroprocesora. Pomoću njega se izvršavaju aritmetičke i logičke operacije nad podacima. U njega se podaci mogu preneti iz memorijske lokacije ili registra, i obrnuto.

Akumulator je osmobitni registar, što znači da je podatak koji se u njemu nalazi brojna vrednost između 0 i 255.

X—Indeksni registar X

Indeksni registar X (engl. X index register) je osmobitni registar kojim se mogu ostvariti skoro sve operacije kao i sa akumulatorom. X registar, kako se skraćeno označava ovaj registar, uvodi nove operacije i proširuje mogućnosti prebacivanja i transformacije podataka.

Y—Indeksni registar Y

Indeksni registar Y (engl. Y index register) je osmobitni registar za koji, u pogledu onoga što pruža, važi navedeno za X registar. Određene razlike postoje između ova dva registra. Skraćeno se označava sa Y registar.

PC — Programski brojač

Programski brojač (engl. program counter) je 16-bitni registar posebne namene koji sadrži adresu memorijske lokacije iz koje se preuzima kôd naredbe koja treba da se izvrši.

Sadržaj PC registra automatski se povećava nakon što je procesor učitao kôd odgovarajuće naredbe. Naredbe skoka postavljaju nove vrednosti u PC registar. Te nove vrednosti su adrese od kojih se nastavlja izvršavanje programa.

SP — Pokazivač steka

Pokazivač steka (engl. stack pointer) je osmobitni registar koji sadrži adresu prve slobodne memorijske lokacije u delu RAM memorije koji se naziva stek. Stek služi za privremeno smeštanje podataka i adresa (npr. povratna adresa pri odlasku u potprogram).

Izvršavanjem odgovarajuće mašinske naredbe podatak se iz određenog registra stavlja na stek ili sa steka vraća u jedan od registra.

Stek se u Komodoru nalazi od adrese 256 ($100) do 511 ($1FF), a popunjava se od viših ka nižim adresama.

P — Rregistar stanja procesora

Registar stanja procesora (engl. status register) treba posmatrati kao grupu od 8 bita od kojih se koristi 7. Svaki od tih 7 bita je indikator (engl. flag) nekog stanja do koga se došlo u toku rada procesora. To može biti na primer da li je rezultat neke operacije pozitivan, negativan ili jednak nuli.

DDR — Registar smera prenosa podataka

Registar smera prenosa podataka (engl. data direction register) je memorijska lokacija na adresi 0. Ona svojom vrednošću određuje smer prenosa podataka preko registra za prenos podataka (DR).

DR — Registar za prenos podataka

Registar za prenos podataka (engl. data register) je memorijska lokacija na adresi 1. Preko svakog njegovog bita se prenose podaci. Da li je određeni bit ulazni ili izlazni određeno je stanjem odgovarajućeg bita registra smera podataka (DDR).

Registri DDR i DR koriste se za kontrolu memorije i rada kasetofona, a detaljnije su određeni u poglavljima 8 i 11.

Razlika između mikroprocesora 6510 i njegovog prethodnika 6502 je u tome što procesor 6502 ne poseduje DDR i DR registre.

Aritmetičko logička jedinica

Ova jedinica (engl. arithmetic and logic unit) je deo mikroprocesora koji omogućava da se nad podacima obave sledeće aritmetičke i logičke operacije:

— sabiranje i oduzimanje,
— povećanje i smanjenje za jedan,
— logičko I, ILI i isključivo ILI,
— poređenje,
— aritmetička pomeranja i rotacije,
— postavljanje, testiranje i brisanje bita.

Kontrolna jedinica

Kôd svake naredbe se čita iz memorije i upisuje u registar naredbi mikroprocesor (engl. instruction register). Kontrolna jedinica (engl. CPU control) dekoduje kôd naredb i na osnovu njega generiše potrebne unutrašnje i spoljašnje signale koji su potrebni da bi s željena operacija obavila.

7.4 NAČINI ADRESIRANJA

Naredbe mikroprocesora 6510 odnose se na prebacivanje i transformaciju podataka koji se nalaze u njegovim internim registrima, spoljnoj memoriji ili perifernim uređajima. Na ovom mestu treba se samo upoznati sa postojećim načinima adresiranja, a njihovo potpuno razumevanje će uslediti tokom upoznavanja sa naredbama.

Akumulatorsko adresiranje (accumulator)

Operacija se izvršava samo na sadržaju akumulatora. Naredba je jednobajtna, a vrednost tog bajta je kôd same operacije. Na primer naredba koja izvršava logičko pomeranje udesno bitova akumulatora **(LSR A)** predstavljena je bajtom vrednosti $4A (znak $ ukazuje da je broj u heksadecimalnom brojnom sistemu).

Postoje 4 naredbe u kojima se primenjuje ovaj način adresiranja.

Implicitno adresiranje (implied)

Operacija se izvršava nad sadržajem nekog od registara. Naredba je dužine jedan bajt, a njegova vrednost je kôd operacije. Na primer punjenje akumulatora sadržajem registra Y (naredba **TYA)** predstavljeno je jednim bajtom (vrednost $98), ili na primer ciklus čekanja ($EA) tj. neizvršavanja ni jedne operacije (naredba **NOP)** samo povećava sadržaj programskog brojača (registar PC).

Postoji 25 naredbi u kojima se primenjuje ovaj način adresiranja.

Neposredno adresiranje (immediate)

Podatak nad kojim se izvršava operacija nalazi se u bajtu koji sledi kôd operacije. Naredba je dvobajtna, prvi bajt je kôd operacije, a drugi operand. Primer naredbe u kojoj je upotrebljen neposredni način adresiranja je punjenje akumulatora određenom brojnom vrednošću tj. konstantom. U naredbi punjenja akumulatora sa vrednošću 255 **(LDA #$FF)**, kôd operacije je $A9 (znak # je rezervisan i obavezan u označavanju operanda u neposrednom adresiranju), a operand je $FF.

Postoji 11 naredbi u kojima se primenjuje ovaj način adresiranja.

Adresiranje nulte strane (zero page)

Podatak nad kojim se izvršava operacija nalazi se na nultoj strani (memorija od adrese 0 do adrese 255 ($FF)). Za njegovo adresiranje dovoljan je jedan bajt. To je drugi bajt u naredbi koja ima mogućnost adresiranja nulte strane. Naredba je dvobajtna, a prvi bajt je kôd operacije. Na primer, naredba punjenja memorijske lokacije 254 ($FE) sadržajem registra X **(STX 254)** predstavljena je sa dva bajta. Prvi bajt ima vrednost $86, a drugi $FE. Prednosti obrade podataka na nultoj strani su veća brzina izvršavanja i manja potrošnja memorije.

Postoji 21 naredba u kojoj se primenjuje ovaj način adresiranja.

Adresiranje nulte strane indeksirano registrom X (zero page indexed by X)

Drugi bajt naredbe, plus sadržaj registra X (bez bita prenosa C) je adresa na nultoj strani na kojoj se nalazi podatak nad kojim se izvršava operacija. Naredba je dvobajtna, a prvi bajt je kôd operacije.

Postoji 16 naredbi u kojima se primenjuje ovaj način adresiranja.

Adresiranje nulte strane indeksirano registrom Y (zero page indexed by Y)

Drugi bajt naredbe, plus sadržaj registra Y (bez bita prenosa C) je adresa na nultoj strani na kojoj se nalazi podatak nad kojim se izvršava operacija. Naredba je dvobajtna, a prvi bajt je kôd operacije.

Postoje 2 naredbe u kojima se primenjuje ovaj način adresiranja.

Aposlutno (direktno) adresiranje (absolute)

Adresa podatka nad kojim se izvršava operacija je u drugom i trećem bajtu naredbe. Naredba je trobajtna, prvi bajt je kôd operacije, drugi je niži bajt adrese, a treći je viši bajt adrese. Na primer, punjenje akumulatora sadržajem memorijske lokacije 49152 ($C000) predstavlja se sa tri bajta od kojih prvi ima vrednost $AD (kôd operacije), drugi vrednost nula i treći vrednost $C0.

Postoji 23 naredbe u kojima se primenjuje ovaj način adresiranja.

Apsolutno adresiranje indeksirano registrom X (absolute indexed by Y)

Adresa u drugom i trećem bajtu naredbe, plus sadržaj registra X je adresa na kojoj se nalazi podatak nad kojim se izvršava naredba. Naredba je trobajtna, a prvi bajt je kôd operacije.

Postoji 15 naredbi u kojima se primenjuje ovaj način adresiranja.

Apsolutno adresiranje indeksirano registrom Y (absolute indexed by Y)

Adresa u drugom i trećem bajtu naredbe, plus sadržaj registra Y je adresa na kojoj se nalazi podatak nad kojim se izvršava naredba. Naredba je trobajtna, a prvi bajt je kôd operacije.

Postoji 9 naredbi u kojima se primenjuje ovaj način adresiranja.

Indirektno adresiranje preindeksirano registrom X (indirect pre-indexed by X)

Drugi bajt naredbe, plus sadržaj registra X (bez bita prenosa C) daje adresu na nultoj strani na kojoj se nalazi dvobajtna adresa podatka. Naredba je dvobajtna, a prvi bajt je kôd operacije. Na primer, ako je sadržaj registra X jednak 3, naredba punjenja akumulatora **LDA (5,X)** daje adresu 8. Akumulator će se napuniti vrednošću koja se nalazi na adresi koja je određena sadržajem lokacija 8 i 9.

Postoji 8 naredbi u kojima se primenjuje ovaj način adresiranja.

Indirektno adresiranje postindeksirano registrom Y (indirect post-indexed by Y)

Sadržaj dvobajtne adrese na nultoj strani određene drugim bajtom naredbe, plus sadržaj Y registra je adresa na kojoj se nalazi podatak nad kojim se izvršava operacija. Naredba je dvobajtna, a prvi bajt je kôd operacije.

Postoji 8 naredbi u kojima se primenjuje ovaj način adresiranja.

Relativno adresiranje (relative)

Drugi bajt u naredbi sadrži udaljenost (engl. offset), u bajtima, memorijske lokacije od koje će se nastaviti izvršavanje programa. Relativno adresiranje se odnosi na naredbe kratkog skoka (do 126 memorijskih lokacija unapred i do 129 memorijskih lokacija unazad). Naredba je dvobajtna, a prvi bajt je kôd operacije.

Postoji 8 naredbi u kojima se primenjuje ovaj način adresiranja.

Indirektni skok (indirect jump)

Samo jedna naredba, naredba indirektnog skoka, koristi ovaj način adresiranja. Pri tome adresa u drugom i trećem bajtu naredbe je adresa na kojoj se nalazi adresa od koje će se nastaviti program.

7.5 NAREDBE MIKROPROCESORA 6510

Mikroprocesor 6510 može izvršiti ukupno 56 naredbi (151 različitu operaciju zavisno od načina adresiranja) koje se mogu podeliti u 5 grupa:

1. naredbe premeštanja podataka (transfer, load and store)
2. naredbe steka (stack operations)
3. aritmetičke i logičke naredbe (arithmetic and logic)
4. naredbe uslovnog grananja, skoka i povratka (branch, jump and return)
5. naredbe kontrole procesora (CPU control)

7.5.1 Naredbe premeštanja podataka

Naredbe premeštanja prebacuju podatke interno između registara procesora ili između registara i spoljne memorije.

Primer za interno premeštanje je prebacivanje sadržaja akumulatora u registar X. Primer za premeštanje podatka između registra i spoljne memorije je prebacivanje sadržaja memorijske lokacije NN u akumulator. Pri tome NN označava bilo koju memorijsku lokaciju od 0 do 65535.

Osnovna karakteristika naredbi premeštanja je da se sadržaj mesta sa koga se podatak uzima ne menja.

Naredbe premeštanja se mogu podeliti u tri grupe:

– premeštanje sadržaja jednog registra u drugi (transfer)
– premeštanje podataka iz memorije u registre (load)
– premeštanje podataka iz registara u memorijske lokacije (store)

Naredbe internog premeštanja sadržaja registara ostvaruju prebacivanje sadržaja jednog registra u drugi. Jedna od njih je, na primer, premeštanje sadržaja registra A u registar Y. Izvršenjem naredbe vrednost u registru Y je jednaka vrednosti u registru A.

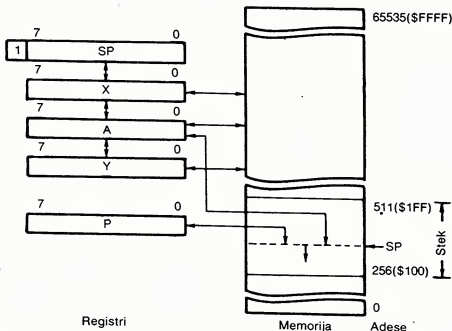
Ukupno postoji 6 naredbi ove grupe. To su sledeće naredbe:

TAX premeštanje sadržaja registra A u registar X
TXA premeštanje sadržaja registra X u registar A
TAY premeštanje sadržaja registra A u registar Y
TYA premeštanje sadržaja registra Y u registar A
TSX premeštanje sadržaja registra S u registar X
TXS premeštanje sadržaja registra X u registar S

Mnemonička oznaka ovih naredbi započinje slovima T, što je skraćenica od engleske reči transfer (premestiti). Naredbe ove grupe su dužine jedan bajt i u njemu je kôd operacije.

Uočavaju se najveće mogućnosti akumulatora (registar A) i registra X u internom premeštanju podataka. Akumulator može razmeniti svoj sadržaj sa registrima X. i Y, a registar X sa akumulatorom i pokazivačem steka (registar S). Na slici 7.2 vizuelno je prikazano,

strelicama između registara, između kojih registara je moguće premeštati podatke. Na slici je prikazana i razmena podataka sa memorijom i registrom P, što će biti objašnjeno u daljem tekstu.



Sl. 7. 2. Prikaz razmene podataka između registara i memorijskih lokacija

Naredbe premeštanja podataka u registre ostvaruju punjenje registara podacima, neposredno ili iz memorijskih lokacija. Neposredno punjenje registara podacima je na primer punjenje registra A konstantom 5. Punjenje, tj. premeštanje podataka iz memorije je na primer punjenje registra X sadržajem memorijske lokacije 1024.

U ovim premeštanjima podataka mogu učestvovati samo registri A, X i Y, što je vidljivo sa slike 7.2. Naredbe ove grupe su:

LDA punjenje akumulatora podatkom
LDX punjenje registra X podatkom
LDY punjenje registra Y podatkom

Mnemonička oznaka ovih naredbi počinje slovima **LD** što je skraćenica od engleske reči load (napuniti). Spisak mnemoničkih oznaka, u zavisnosti od načina adresiranja, dat je u spisku naredbi (poglavlje 7.7). Neposredno punjenje registra koristi različito označavanje. Ispred operanda, brojne vrednosti, treba navesti znak povisilice (#). Time je označeno da je brojna vrednost podatak, a ne adresa memorijske lokacije sa čijom vrednošću se puni registar. Na primer naredba:

LDA #78

označava da se akumulator puni vrednošću 78 (decimalno), a naredba:

LDA 78

označava da se akumulator puni sadržajem memorijske lokacije 78.

Naredbe premeštanja podataka u memorijske lokacije premeštaju sadržaje registara u memorijske lokacije. Primer naredbe ove grupe je smeštanje sadržaja registra Y u memorijsku lokaciju 40960 ($A000).

U ovim premeštanjima podataka u memorijske lokacije mogu učestvovati registri A, X i Y, što je vidljivo sa slike 7.2. Naredbe ove grupe su:

STA smeštanje sadržaja akumulatora u memoriju
STX smeštanje sadržaja registra X u memoriju
STY smeštanje sadržaja registra Y u memoriju

Mnemonička oznaka ovih naredbi počinje slovima ST što je skraćenica od engleske reči store (smestiti, sačuvati). Naredbe su dvo ili trobajtne zavisno od načina adresiranja. Na primer naredba:

STA 23

je dužine dva bajta i označava da se sadržaj registra A smešta u memorijsku lokaciju 23 (na nultoj strani), a naredba:

STA $A002

je dužine tri bajta i označava da se sadržaj akumulatora smešta u memorijsku lokaciju A002 heksadecimalno, tj. 40962 decimalno.

Spisak naredbi i načina adresiranja ove, a i drugih grupa je dat u spisku naredbi na kraju ovoga poglavlja.

Naredbe za premeštanje podataka primenjene su i prikazane u svim datim primerima.

7.5.2 Naredbe steka

Stek je deo RAM memorije i služi za privremeno smeštanje podataka. Stek se nalazi od adrese 256 do 511 (binarno 100000000 do 111111111) što znači da se u njega može smestiti do 256 osmobitnih podataka. To može biti ograničenje koje treba imati u vidu pri radu sa stekom.

Naredbama steka na stek se mogu staviti sadržaji registara A i P. Takođe podaci se mogu vratiti u registre A i P (slika 7.2). Stavljanje podataka na stek uvek se obavlja od viših adresa ka nižim adresama na steku. Vraćanje podataka sa steka se obavlja uvek od podataka na nižim adresama ka podacima na višim adresama steka. To znači da je stek organizovan po ptincipu "poslednji unutra-prvi napolje" (LIFO last-in-first-out).

Upotrebom naredbi za stavljanje na stek određeni podatak se stavlja na stek u memorijsku lokaciju adresiranu pokazivačem steka (SP), a sadržaj pokazivača steka (SP) se smanjuje za jedan, ukazujući na sledeću slobodnu memorijsku lokaciju. Izvršenjem naredbe za prebacivanje podataka sa steka u registre sadržaj SP registra se povećava za jedan, a podatak u memorijskoj lokaciji adresiran tim novim sadržajem preslikava se u odgovarajući registar.

Korišćenje steka se sastoji u upotrebi dve naredbe, jedne za stavljanje na stek i druge za vraćanje sa steka. Upotreba steka je izazov za programera, a njegova primena je velika iz razloga što omogućava pisanje programa koji su kraći i koji se brže izvršavaju.

Primeri 1 i 8 prikazuju naredbe steka.

PHA (push A) naredba stavlja sadržaj akumulatora na stek. Sadržaj pokazivača steka je umanjen za jedan, a sadržaj akumulatora i indikatori stanja su nepromenjeni.

PLA (pull A) naredba podatak sa steka stavlja u akumulator. Sadržaj pokazivača steka se povećava za jedan. Indikatori stanja N i Z dobijaju vrednosti u zavisnosti od vrednosti koja se stavlja u akumulator.

PHP (push P) naredba stavlja sadržaj registra stanja procesora na stek. Sadržaj pokazivača steka je umanjen za jedan, a sadržaj registara P i A je nepromenjen.

PLP (pull P) naredba podatak sa steka stavlja u registar stanja procesora. Sadržaj pokazivača steka je povećan za jedan.

7.5.3 Aritmetičke i logičke naredbe

Pomoću naredbi ove grupe obavljaju se aritmetičke operacije sabiranja, oduzimanja, povećanja i umanjenja za jedan, operacije poređenja, logičke operacije sabiranja i množenja kao i aritmetičko logičke operacije manipulacije sa bitima.

Sabiranje i oduzimanje

Mikroprocesor 6510 omogućuje sabiranje i oduzimanje osmobitnih brojnih vrednosti. Pri tome u operaciji sabiranja ili oduzimanja učestvuju sadržaj akumulatora i memorije, a rezultat se smešta u akumulator. U obe operacije učestvuje i indikator prenosa (bit C registra P), koji je zajedno sa drugim indikatorima objašnjen u tački 7.6.

Izvršenjem naredbi, indikatori stanja procesora menjaju vrednosti u zavisnosti od rezultata operacije. Naredbe se mogu koristiti u binarnom ili decimalnom načinu rada u zavisnosti od stanja bita D registra P.

Primer 3 ilustruje naredbu sabiranja i oduzimanja.

ADC (add with carry) naredba sabira sadržaj akumulatora, sadržaj memorijske lokacije i vrednost bita C. Rezultat se smešta u akumulator.

Data je naredba sabiranja sadržaja registra A sa podatkom koji se nalazi u memorijskoj lokaciji 123 i bitom C:

ADC 123

Ako se u akumulatoru nalazi na primer vrednost $E2, u lokaciji 123 na primer vrednost 3, a vrednost bita C je jedan, važi sledeće:

Pre operacije	A	$E2
	sadržaj memorije	$03
	C	1
Posle operacije	A	$E6
	sadržaj memorije	$03
	C	0

U slučaju da je C bit pre izvršenja operacije bio jednak nuli, u akumulatoru bi rezultat bio $E5.

Kada je rezultat sabiranja registara veći od $FF (decimalno 255) događa se sledeće:

Pre operacije	A	$E2
	sadržaj memorije	$1F
	C	1

Posle operacije	A	\$02
	sadržaj memorije	\$1F
	C	1

U slučaju da je C bit pre izvršenja operacije bio jednak nuli, u akumulatoru bi rezultat bio \$01, a C bit bi takođe dobio vrednost 1.

SBC (subtract with carry) oduzima od sadržaja akumulatora sadržaj memorijske lokacije i invertovanu vrednost bita C. Rezultat se smešta u akumulator.

To znači da ako se ne želi uticaj bita C na rezultat oduzimanja potrebno je da vrednost bita C pre oduzimanja bude jednaka jedan (naredba **SEC**).

Data je naredba oduzimanja od sadržaja registra A podatka koj ise nalazi u memorijskoj lokaciji \$4000 (16384 decimalno) i invertovanog bita C:

SBC \$4000

Ako se u akumulatoru nalazi naprimer vrednost \$2F, u lokaciji \$4000 vrednost \$1A, a vrednost bita C je jedan, važi sledeće:

Pre operacije	A	\$2F
	sadržaj memorije	\$1A
	C	1
Posle operacije	A	\$15
	sadržaj memorije	\$1A
	C	1

U slučaju da je C bit pre izvršenja operacije bio jednak nuli, u akumulatoru bi rezultat bio \$14.

Izvršenjem naredbe **SBC**, ako je rezultat veći ili jednak nuli $(A-M \geqslant 0)$, bit prenosa C dobija vrednost 1. Ako je rezultat oduzimanja manji od nule $(A-M<0)$ bit prenosa dobija vrednost 0, a rezultat koji se upisuje u akumulator je: $256-(A-M)$. Na primer:

Pre operacije	A	\$2F
	sadržaj memorije	\$4C
	C	1
Posle operacije	A	\$E3
	sadržaj memorije	\$4C
	C	1

Povećanje i umanjenje za jedan

Naredbe povećanja i smanjenja povećavaju odnosno smanjuju sadržaj memorijskih lokacija i registara X i Y za jedan.

Primeri 4,6,7,8,9 i 10 prikazuju njihovu upotrebu.

INC (increment memory) naredbom povećava se sadržaj memorijske lokacije za jedan $(M=M+1)$. Data je mnemonička oznaka povećanja sadržaja memorijske lokacije određene zbirom brojne vrednosti 5 i sadržajem registra X (adresiranje nulte strane indeksirano registrom X).

INC 5,X

INX (increment X) naredbom povećava se sadržaj registra X za jedan (X=X+1).

INY (increment Y) naredbom povećava se sadržaj registra Y za jedan (Y=Y+1).

DEC (decrement memory) naredbom umanjuje se sadržaj memorijske lokacije za jedan M=M−1).

DEX (decrement X) naredbom umanjuje se sadržaj registra X za jedan (X=X−1).

DEY (decrement Y) naredbom umanjuje se sadržaj registra Y za jedan (Y=Y−1).

Poređenje

Naredbe poređenja upoređuju sadržaj registara A, X i Y sa sadržajem memorijskih lokacija ili neposredno sa brojnim podacima. Poređenje se obavlja oduzimanjem brojnih vrednosti, ali se pri tome ne menjaju sadržaji registara i memorijskih lokacija. Rezultat utiče samo na indikatore N, Z i C zavisno od rezultata operacije:

Z bit dobija vrednost 1 ako su podaci isti (A=M), vrednost 0 ako su različiti (A<>M)
N bit dobija vrednost bita 7 rezultata, tj. postaje 1 ako je rezultat negativan, a 0 ako je rezultat pozitivan
C bit dobija vrednost 1 ako je sadržaj registra jednak ili veći od podatka sa kojim se poredi (A>M), u protivnom dobija vrednost 0.

Naredbe ove grupe su uobičajeno praćene naredbama grananja. Time se zavisno od rezultata poređenja određuje dalji tok programa.
Naredbe ove grupe prikazane su primerima: 4, 9 i 10.

CMP (compare to accumulator) naredba upoređuje sadržaj akumulatora sa naznačenim podatkom. Od sadržaja akumulatora oduzima se brojna vrednost podatka. Sadržaj akumulatora se ne menja, a indikatori N, Z i C dobijaju vrednosti zavisno od rezultata.

CPX (compare to X) naredba upoređuje sadržaj registra X sa naznačenim podatkom. Od sadržaja registra X oduzima se brojna vrednost podatka. Sadržaj registara se ne menja, a indikatori N, Z i C dobijaju vrednosti zavisno od rezultata.

CPY (compare to Y) naredba upoređuje sadržaj registra Y sa naznačenim podatkom. Od sadržaja registra Y oduzima se brojna vrednost podatka. Sadržaj registara se ne menja, a indikatori N, Z i C dobijaju vrednosti zavisno od rezultata. Data je mnemonička oznaka poređenja sadržaja registra Y i brojne vrednosti 10.

CPY #10

Logičke operacije

Logičke operacije, tzv. Bulove operacije, su operacije logičkog množenja, logičkog sabiranja i isključivog (ekskluzivnog) sabiranja. Izvršavaju se između sadržaja akumulatora i naznačenog podatka. Rezultat se smešta u akumulator.
Primer 5 ilustruje naredbe ove grupe.

AND (logical and) naredba izvršava logičko množenje (I) sadržaja akumulatora i naznačenog podatka. Operacija se izvodi bit po bit, a rezultat se smešta u akumulator.
U sledećem primeru se operacija izvodi između sadržaja akumulatora i sadržaja memorijske lokacije 50000:

AND 50000

akumulator	11010010
sadržaj memorijske lokacije	10000111
akumulator posle operacije	10000010

Uočava se da su izvršenjem naredbe neki bitovi akumulatora promenili vrednost iz 1 u 0. To se često koristi u postupku koji prevodi vrednosti određenih bita iz 1 u 0, i koji se zove maskiranje.

ORA (logical or) naredba izvršava logičko sabiranje (ILI) sadržaja akumulatora i naznačenog podatka. Operacija se izvodi bit po bit, a rezultat se smešta u akumulator.

U sledećem primeru se operacija izvodi između sadržaja akumulatora i sadržaja memorijske lokacije određene zbirom vrednosti 50000 i sadržaja registra X (apsolutno adresiranje indeksirano registrom X):

ORA 50000,X

akumulator	11010010
sadržaj memorijske lokacije	10000111
akumulator posle operacije	11010111

Uočava se da su izvršenjem naredbe neki bitovi akumulatora promenili vrednost iz 0 u 1. To se često koristi u postupku za prevođenje vrednosti određenih bita iz 0 u 1.

EOR (exclusive or) naredba izvršava logičko isključivo sabiranje (isključivo ILI) sadržaja akumulatora i naznačenog podatka. Operacija se izvodi bit po bit. Operacija između dve binarne cifre daje rezultat jedinicu ako te cifre nisu iste. Rezultat se smešta u akumulator.

U sledećem primeru se operacija izvodi između sadržaja akumulatora i broja 87 (heksadecimalno).

EOR # $87

akumulator	11010010
brojna vrednost	10000111
akumulator posle operacije	01010101

Ova naredba se najčešće koristi za invertovanje bita. To se postiže tako što se izvrši operacija između podatka koji se želi invertovati i broja $FF. Time se dobija komplement polaznog podatka.

Manipulacija bitima

BIT (test bit) naredba omogućuje da procesor 6510 izvrši proveru vrednosti bita 6 i bita 7 memorijskih lokacija.

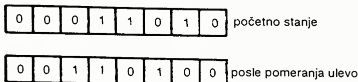
Izvršenjem naredbe ostvaruje se logičko množenje **(AND)** između sadržaja akumulatora i memorijske lokacije. Akumulator i memorijska lokacija zadržavaju svoju vrednost, a u zavisnosti od rezultata postavlja se indikator Z. Uz to bitovi 6 i 7 memorijske lokacije se premeštaju u bite V i N registra stanja procesora P.

Ova naredba se može upotrebiti adresiranjem nulte strane i apsolutnim adresiranjem, a koristi se obično za testiranje registara perifernih uređaja.

Pomeranje i rotacija

Naredbe ove grupe premeštaju bite u akumulatoru ili memorijskim lokacijama za po jedno mesto u levo ili u desno.

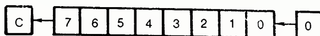
To može biti od značaja kada se uzme u obzir da se pomeranjem bita za jedno mesto ulevo vrši množenje sa dva, a pomeranjem u desno postiže se deljenje sa dva. Primer množenja sa 2, vrednosti sadržane u jednoj memorijskoj lokaciji prikazan je na slici 7.3.



Al. 7. 3. Prikaz operacije pomeranja bita u levo

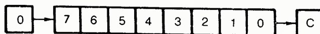
Naredbe ove grupe utiču na indikatore N, Z i C, i zbog toga su od značaja u određivanju toka programa. Mogućnosti primene naredbi pomeranja i rotacije su velike, a neke od njih su prikazane u primerima 6 i 8.

ASL (arithmetic shift left) naredba pomera sadržaj akumulatora ili memorijske lokacije u levo za jednu lokaciju bita. Bit 7 se premešta u bit C, a bit nula dobija vrednost 0. Rezultat se nalazi u polaznoj lokaciji, akumulatoru ili memorijskoj lokaciji.



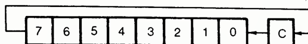
Slika 7. 4. Prikaz operacije aritmetičkog pomeranja u levo

LSR (logical shift right) naredba pomera sadržaj akumulatora ili memorijske lokacije u desno za jednu lokaciju bita. Bit 0 se premešta u bit C, a bit sedam dobija vrednost 0. Rezultat se nalazi u polaznoj lokaciji, akumulatoru ili memorijskoj lokaciji.



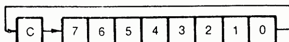
Slika 7. 5. Prikaz operacije logičkog pomeranja u desno

ROL (rotate left) naredba rotira sadržaj akumulatora ili memorijske lokacije u levo za jednu lokaciju bita. C bit se premešta u bit 0, a bit 7 u bit C određujući mu novu vrednost. U rotaciji učestvuje 9 bita.



Sl 7. 6. Prikaz operacije rotacije u levo

ROR (rotate right) naredba rotira sadržaj akumulatora ili memorijske lokacije u desno za jednu lokaciju bita. C bit se premešta u bit 7, a bit 0 u bit C određujući mu novu vrednost. U rotaciji učestvuje 9 bita.



Sl. 7 7. Prikaz operacije rotacije u desno

Data je mnemonička oznaka rotacije u desno sadržaja akumulatora:

ROR A

7.5.4 Naredbe uslovnog grananja, skoka i povratka

Naredbe ove grupe menjaju tok izvršavanja programa. Kao i u višim programskim jezicima veoma su značajne i često korišćene.

Naredbe uslovnog grananja

Naredbe uslovnog grananja menjaju tok izvršavanja programa ako je postavljeni uslov zadovoljen. Na primer, ako je rezultat prethodne operacije jednak nuli izvršavanje programa će se nastaviti od naznačene adrese.

Uslovi koji određuju da li se program nastavlja izvršenjem naredbe u sledećoj memorijskoj lokaciji ili se nastavlja od naredbe u nekoj drugoj memorijskoj lokaciji su:

- grananje u zavisnosti od prenosa (indikator C)
- grananje u zavisnosti od toga da li je rezultat jednak ili različit od nule (indikator Z)
- grananje u zavisnosti od znaka rezultata (indikator N)
- grananje u zavisnosti od premašenja (indikator V)

Ako je postavljeni uslov ispunjen izvršiće se grananje na naznačenu memorijsku lokaciju.

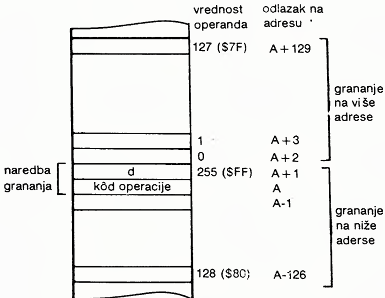
Naredbe uslovnog grananja su dvobajtne, a način adresiranja je relativni. To znači da je u prvom bajtu kôd operacije, a u drugom operand koji određuje udaljenost (engl. displacement) memorijske lokacije sa koje se nastavlja dalje izvršavanje programa. S obzirom da je operand jednobajtni i da se sedmi bit operanda koristi za određivanje znaka, naredbe ove grupe omogućuju nastavljanje programa od adrese koja je do 127 veća i do 128 manja od adrese lokacije koja sledi naredbu uslovnog grananja.

Grananje se obavlja tako što se izvršenjem naredbe sadržaju programskog brojača doda vrednost operanda (po konceptu binarnih brojeva u komplementu dvojke). Na slici 7.8 je simbolički predstavljen deo memorije u okviru koga se obavlja uslovno grananje za različite vrednosti 8-bitnog operanda d.

Sa slike se uočava da moguće vrednosti operanda od −128 do +127 efektivno daju udaljenosti od −126 do +129 od adrese na kojoj se nalazi naredba grananja. Pri grananju unapred, na više adresne lokacije, operand d se može smatrati brojem preskočenih lokacija. Na primer ako operand ima vrednost 2 program će se nastaviti, ako je uslov ispunjen, od lokacije koja je po adresi veća za 2 od lokacije koja sledi naredbu grananja.

Pri korišćenju programa asemblera za pisanje mašinskih programa otpada potreba za preračunavanjem udaljenosti u operand. Označavanjem adresa simboličkim oznakama-labelama asembler određuje i dodeljuje vrednosti operandima.

Naredbe uslovnog grananja nemaju uticaj na stanje indikatora tj. na registar stanja procesora.



Sl. 7. 8. Adrese pri naredbama uslovnog grananja

Naredbe uslovnog grananja su vrlo praktične i korišćene naredbe koje omogućavaju kratke skokove u programu. Obično im prethode naredbe poređenja, testiranja ili logičke operacije pomoću kojih se obavlja postavljanje indikatora.

Naredbe ove grupe su ilustrovane primerima 4, 6, 8, 9 i 10.

BCC (branch if carry clear) naredbom program se nastavlja od naznačene adrese ako je bit C=0.

BCS (branch if carry set) naredbom program se nastavlja od naznačene adrese ako je bit C=1.

BEQ (branch if equal to zero) naredbom program se nastavlja od naznačene adrese ako je rezultat prethodne operacije nula, tj. ako je bit Z=1.

BNE (branch if not equal to zero) naredbom program se nastavlja od naznačene adrese ako je rezultat prethodne operacije različit od nule, tj. ako je bit Z=0.

BMI (branch if minus) naredbom program se nastavlja od naznačene adrese ako je rezultat prethodne operacije manji od nule, tj. ako je bit N=1.

BPL (branch if plus) naredbom program se nastavlja od naznačene adrese ako je rezultat prethodne operacije veći ili jednak nuli tj. ako je bit N=0.

BVC (branch if overflow clear) program se nastavlja od naznačene adrese ako ne postoji premašenje tj. ako je bit V=0.

BVS (branch if overflow set) program se nastavlja od naznačene adrese ako postoji premašenje tj. ako je bit V=1.

Data je mnemonička oznaka naredbe kojom će se program nastaviti, ako postoji premašenje, od memorijske lokacije koja je po adresi za 10 veća od adrese koja sledi naredbu:

BVS 10

Naredbe skoka

Naredbe skoka bezuslovno preusmeravaju tok programa.

U toku izvršavanja programa, sadržaj programskog brojača PC se povećava izvršavanjem svake naredbe. Pri tome njegov sadržaj adresira memorijsku lokaciju u kojoj je smeštena sledeća naredba koja treba da se izvrši. Izvršenjem naredbi skoka, programski brojač se postavlja, ne na sledeću, već na novu adresu od koje će se nastaviti izvršavanje programa.

Naredbe skoka ne menjaju vrednosti indikatora, tj. registar stanja procesora (P).

Naredbe skoka su ilustrovane primerima 7 i 10.

JMP (jump to address) naredbom programski brojač se puni novom adresom, što će rezultovati nastavkom programa od te adrese. Naredba odgovara bejzik **GOTO** naredbi.

Adresiranje nove adrese može biti apsolutno ili indirektno. Data je mnemonička oznaka skoka (indirektno adresiranje) na adresu koja je određena sadržajem memorijskih lokacija $FFFE (bajt manje težine) i $FFFF (bajt veće težine).

JMP ($**FFFE**)

JSR (jump to subroutine). Pre punjenja programskog brojača novom adresom stavlja se na stek sadržaj programskog brojača povećanog za dva. Punjenjem dve lokacije stek memorije adresom naredbe koja sledi naredbu **JSR** sačuvana je povratna adresa. Na taj način je dobijena naredba koja omogućava izvršavanje mašinskih potprograma. Za povratak u glavni program koristi se naredba povratka **RTS.** Naredba **JSR** odgovara bejzik **GOSUB** naredbi.

Adresiranje u ovoj naredbi može biti samo apsolutno. Data je mnemonička oznaka odlaska na potprogram na adresi 30000.

JSR 30000

Naredbe povratka

Naredbe ove grupe ostvaruju vraćanje u programski brojač prethodno sačuvanih vrednosti. Time se ostvaruje povratak iz potprograma i povratak iz programskog prekida.

Naredbe povratka su posebno ilustrovane primerima 7 i 10.

RTS (return from subroutine) naredba koristi se za povratak iz potprograma koji su pozvani naredbom **JSR.** Odgovara bejzik naredbi **RETURN.**

Naredba svojim izvršavanjem puni programski brojač sa dvobajtnom vrednošću sa steka i zatim povećava tu vrednost za jedan. Takođe povećava vrednost pokazivača steka SP za dva (SP=SP+2), čime pokazivač steka ukazuje na novu slobodnu lokaciju stek memorije.

Naredba ne menja vrednosti indikatora.

RTI (return from interrupt) naredba koristi se za povratak iz rutine za obradu programskog prekida koji je nastao izvršenjem naredbe prekida BRK ili pojavom signala IRQ ili NMI (videti poglavlja 8 i 11).

Naredba svojim izvršavanjem puni registar stanja procesora P sa jednobajtnom vrednošću sa steka, a zatim puni programski brojač sa dvobajtnom vrednošću sa steka. Pokazivač steka po izvršenju naredbe ima novu vrednost (SP=SP+3) i ukazuje na novu slobodnu lokaciju na stek memoriji.

7.5.5 Naredbe kontrole procesora

U ovoj grupi naredbi nalazi se preostalih devet naredbi procesora. To su kontrolne naredbe opšte namene i one utiču na stanje procesora.

NOP (no operation) naredba ne izvršava ništa. Može se upotrebiti za formiranje kašnjenja. Po izvršenju programski brojač je uvećan za jedan.

BRK (break) naredba izaziva programski prekid. Sadržaj programskog brojača se stavlja na stek (u dva bajta stek memorije), a zatim se sadržaj registra stanja procesora (P) stavlja na stek (jedan bajt). Sadržaji memorijskih lokacija $FFFE i $FFFF se premeštaju u niži i viši bajt programskog brojača respektivno. Bit B registra P je dobio vrednost 1 pre stavljanja na stek, da bi se označilo da je do programskog prekida došlo izvršenjem naredbe **BRK**. Programski prekid može izazvati i pojava signala prekida IRQ i NMI na odgovarajućim izvodima mikroprocesora (videti poglavlje 11) pri čemu bit B zadržava vrednost 0.

Pojavom programskog prekida obustavlja se izvršavanje tekućeg programa i prelazi se na izvršavanje programa koji je predviđen za tu svrhu (program za obradu prekida). Posle njegovog izvršavanja nastavlja se izvršavanje prekinutog programa. U radu sa procesorom 6510 moguća su tri uzroka prekida: softverski prekid (naredba **BRK**), hardverski prekid koji se može onemogućiti (IRQ — interrupt request) i hardverski prekid koji se ne može onemogućiti (NMI — non maskable interrupt). Onemogućenje prekida se ostvaruje dodeljivanjem vrednosti 1 bitu I registra P.

CLI (clear interrupt disable) naredba dodeljuje bitu I vrednost 0. Time je omogućeno, pojavom prekida, izvršenje programa za obradu prekida.

SEI (set interrupt disable) naredba dodeljuje bitu I vrednost 1. Time je onemogućen programski prekid (osim u slučaju pojave signala NMI). Koristi se u toku izvršavanja programa za obradu prekida (da ne bi ponovo došlo do istog programskog prekida) i u aplikacijama u radu u realnom vremenu.

CLC (clear carry) naredba dodeljuje bitu C vrednost 0.

SEC (set carry) naredba dodeljuje bitu C vrednost 1.

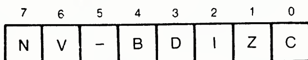
CLD (clear decimal mode) naredba dodeljuje bitu D vrednost 0, čime je postavljen binarni način rada za naredbe **ADC** i **SBC.**

SED (set decimal mode) naredba dodeljuje bitu D vrednost 1, čime je postavljen decimalni način rada za naredbe **ADC** i **SBC.**

CLV (clear overflow) naredba dodeljuje bitu V vrednost 0.

7.6 INDIKATORI STANJA

P registar je registar stanja procesora koga treba posmatrati kao grupu od 8 bita. Vrednosti bita ukazuju na stanje procesora do koga se došlo u toku izvršavanja programa. Raspored bita u registru i oznaka svakog bita je data na slici 7.9.



Sl. 7. 9. Registar stanja procesora

Namena svakog od bita je sledeća (bit 5 se ne koristi):

Bit C je indikator prenosa (engl. carry flag). Sabiranjem dva osmobitna broja rezultat može izaći iz opsega osmobitnih brojeva. Tada se javlja potreba za prenosom u dodatni bit. C bit je taj bit koji će dobiti vrednost 1. Isto se događa i u slučaju oduzimanja, ako rezultat izađe iz opsega osmobitnih brojeva. C bit se može postaviti na jedinicu i pri izvršavanju naredbi rotacija i pomeranja. Vrednost 1 dobija direktno izvršenjem naredbe SEC, vrednost 0 izvršenjem naredbe CLC. U programu stanje C bita se najčešće testira naredbama BCS i BCC.

Naredbe koje utiču na stanje bita C su: ADC, ASL, CLC, CMP, CPX, CPY, LSR, PLP, ROL, ROR, RTI, SBC i SEC.

Bit Z je indikator nule (engl. zero flag). Ovaj bit dobija vrednost 1 samo akoj je rezultat operacije jednak nuli. U programu stanje Z bita se najčešće testira naredbama BEQ i BNE.

Naredbe koje utiču na stanje bita Z su: ADC, AND, ASL, BIT, CMP, CPY, CPX, DEC, DEX, DEY, EOR, INC, INX, INY, LDA, LDX, LDY, LSR, ORA, PLA, PLP, ROL, ROR, RTI, SBC, TAX, TAY, TXA i TYA.

Bit I je indikator onemogućenja programskog prekida (engl. interrupt request disable flag). Ako je vrednost I bita 1, biće onemogućen programski prekid iniciran naredbom BRK ili signalom IRQ na odgovarajućem izvodu mikroprocesora (prekid iniciran signalom NMI nije moguće onemogućiti). Ako je vrednost I bita 0, biće omogućen svaki programski prekid.

Naredbe koje utiču na stanje bita I su: BRK, CLI, PLP, RTI i SEI.

Bit D je indikator decimalnog načina rada (engl. decimal mode flag). Ako je vrednost bita D jednaka 1 procesor prelazi u BCD (binary coded decimal) način rada. U BCD načinu rada decimalne cifre (0 — 9) se predstavljaju sa 4 bita, a decimalni brojevi sa grupama od po 4 bita, gde svaka grupa odgovara jednoj cifri. Na primer decimalni broj 16 se u BCD obliku predstavlja: 00010110, gde prva grupa od 4 bita daje cifru 1, a druga grupa cifru 6. Postavljanjem bita D na jedinicu procesor će brojne vrednosti predstavljati u BCD obliku.

Naredbe koje utiču na stanje D bita su: CLD, PLP, RTI i SED.

Bit B je indikator izvršenja naredbe BRK (engl. break flag). Ovaj bit automatski postavlja procesor na vrednost 1 prilikom izvršenja naredbe BRK. Služi za razlikovanje da li je do prekida došlo programski ili hardverski.

Bit V je indikator premašenja (engl. overflow flag). Koristi se da pokaže da li je rezultat sabiranja ili oduzimanja u komplementu dvojke van opsega od —128 do +127. Ako se to desi javlja se i prenos iz bita 6 u bit 7, i V bit dobija vrednost 1. U sledećem primeru rezultat sabiranja je tačan u apsolutnoj binarnoj formi, ali ne i u formi komplementa dvojke. Bit premašenja se postavlja na jedinicu.

50	00110010
+99	01100011
149	10010101 = −21 u komplementu dvojke

Naredbe koje menjaju stanje bita V su: ADC, BIT, CLV, PLP, RTI i SBC.

Bit N je indikator negativnog znaka (engl. negativ result flag). N bit je u većini slučajeva bit 7 akumulatora. Kada je njegova vrednost 1, ukazuje da je rezultat operacije negativan u formi komplementa dvojke. Praktično bit N je bit 7 rezultata.

Naredbe koje menjaju vrednost bita N su: ADC, AND, ASL, BIT, CMP, CPX, CPY, DEC, DEX, DEY, EOR, INC, INX, INY, LDA, LDX, LDY, LSR, ORA, PLA, PLP, ROL, ROR, TAX, TAY, TXS, TXA i TYA.

7.7 SPISAK NAREDBI MIKROPROCESORA 6510

Naredna tabela prikazuje sve naredbe mikroprocesora 6510. U prvoj koloni je data mnemonička oznaka naredbe i opis njenog dejstva. U drugoj i trećoj su načini adresiranja sa odgovarajućom mnemoničkom oznakom (gde n označava jednobajtni operand, a nn dvobajtni operand). U četvrtoj koloni je broj bajta po naredbi. U petoj koloni je heksadecimalni kôd operacije (prvi bajt naredbe). U poslednjoj koloni je dato kako su postavljeni indikatori u registru P nakon izvršenja naredbe.

Upotrebljene oznake u šestoj koloni su sledeće:

+ Vrednost indikatora se menja u zavisnosti od rezultata naredbe
0 Indikator se postavlja na vrednost 0
1 Indikator se postavlja na vrednost 1
6 Indikator dobija vrednost bita 6 memorijske lokacije
7 Indikator dobija vrednost bita 7 memorijske lokacije

Kada u tabeli nije navedena ni jedna od oznaka indikator ostaje nepromenjen po izvršenju naredbe.

ADC

```
                                                                NV-BDIZC
Dodavanje sadržaja        Neposredno         ADC #n     2   69  ++    ++
memorije i bita C         Nulta strana       ADC n      2   65  ++    ++
sadržaju akumulatora      Nulta strana,X     ADC n,X    2   75  ++    ++
A=A+M+C                   Apsolutno          ADC nn     3   6D  ++    ++
                          Apsolutno,X        ADC nn,X   3   7D  ++    ++
                          Apsolutno,Y        ADC nn,Y   3   79  ++    ++
                          Preindeksirano     ADC (n,X)  2   61  ++    ++
                          Postindeksirano    ADC (n),Y  2   71  ++    ++
```

AND

```
                                                                NV-BDIZC
Logičko množenje (I)      Neposredno         AND #n     2   29  +     +
sadržaja akumulatora      Nulta strana       AND n      2   25  +     +
i memorije                Nulta strana,X     AND n,X    2   35  +     +
A=A AND M                 Apsolutno          AND nn     3   2D  +     +
                          Apsolutno,X        AND nn,X   3   3D  +     +
                          Apsolutno,Y        AND nn,Y   3   39  +     +
                          Preindeksirano     AND (n,X)  2   21  +     +
                          Postindeksirano    AND (n),Y  2   31  +     +
```

ASL

					NV-BDIZC
Pomernje bita u bajtu ulevo za jedno mesto (memorije ili reg. A)	Akumulatorsko	ASL A	1	0A	+ ++
	Nulta strana	ASL n	2	06	+ ++
	Nulta strana,X	ASL n,X	2	16	+ ++
C ← 76543210 ← 0	Apsolutno	ASL nn	3	0E	+ ++
	Apsolutno,X	ASL nn,X	3	1E	+ ++

BCC

					NV-BDIZC
Grananje ako je C=0	Relativno	BCC n	2	90	

BCS

					NV-BDIZC
Grananje ako je C=1	Relativno	BCS n	2	B0	

BEQ

					NV-BDIZC
Grananje ako je Z=1 (ako je rezultat 0)	Relativno	BEQ n	2	F0	

BIT

					NV-BDIZC
Provera bite reg. A i memorijske lokacije A AND M,M7 → N,M6 → V	Nulta strana	BIT n	2	24	76 +
	Apsolutno	BIT nn	3	2C	76 +

BMI

					NV-BDIZC
Grananje ako je N=1 (rezultat negativan)	Relativno	BMI n	2	30	

BNE

					NV-BDIZC
Grananje ako je Z=0 (rezultat nije nula)	Relativno	BNE n	2	D0	

BPL

					NV-BDIZC
Grananje ako je N=0 (rezultat pozitivan ili jednak nuli)	Relativno	BPL n	2	10	

BRK

					NV-BDIZC
Programski prekid PC i P idu na stek	Implicitno	BRK	1	00	1 1

BVC

					NV-BDIZC
Grananje ako je V=0 (nije premašenje u rezultatu)	Relativno	BVC n	2	50	

BVS

					NV-BDIZC
Grananje ako je V=1 (premašenje u rezultatu)	Relativno	BVS n	2	70	

CLC

					NV-BDIZC
Resetovanje bita C C=0	Implicitno	CLC	1	18	0

CLD

					NV-BDIZC
Resetovanje bita D D=0	Implicitno	CLD	1	D8	0

CLI

					NV-BDIZC
Resetovanje bita I I=0	Implicitno	CLI	1	58	0

CLV					NV-BDIZC
Resetovanja bita V V-0	Implicitno	CLV	1	B8	0
CMP					NV-BDIZC
Poredjenja sadržaja	Neposredno	CMP #n	2	C9	+ ++
akumulatora i	Nulta strana	CMP n	2	C5	+ ++
memorijska lokacija	Nulta strana,X	CMP n,X	2	D5	+ ++
	Apsolutno	CMP nn	3	CD	+ ++
	Apsolutno,X	CMP nn,X	3	DD	+ ++
	Apsolutno,Y	CMP nn,Y	3	D9	+ ++
	Preindeksirano	CMP (n,X)	2	C1	+ ++
	Postindaksirano	CMP (n),Y	2	D1	+ ++
CPX					NV-BDIZC
Poradjanja sadržaja	Naposredno	CPX #n	2	E0	+ ++
ragistra X i	Nulta strna	CPX n	2	E4	+ ++
mamorijska lokacija	Apsolutno	CPX nn	3	EC	+ ++
CPY					NV-BDIZC
Poradjanja sadržaja	Neposradno	CPY #n	2	C0	+ ++
ragistra Y i	Nulta strana	CPY n	2	C4	+ ++
mamorisjka lokacije	Apsolutno	CPY nn	3	CC	+ ++
DEC					NV-BDIZC
Umanjenja sadržaja	Nulta strana	DEC n	2	C6	+ +
memorija za jedan	Nulta strana,X	DEC n,X	2	D6	+ +
M-M-1	Apsolutno	DEC nn	3	CE	+ +
	Apsolutno,X	DEC nn,X	3	DE	+ +
DEX					NV-BDIZC
Umanjanja sadržaja rag. X za jadan (X-X-1)	Implicitno	DEX	1	DA	+ +
DEY					NV-BDIZC
Umanjenja sadržaja rag. Y za jadan (Y-Y-1)	Implicitno	DEY	1	88	+ +
EOR					NV-BDIZC
Iskljucivo ILI	Neposradno	EOR #n	2	49	+ +
sadržaja akumulatora	Nulta strana	EOR n	2	45	+ +
i mamorija	Nulta strana,X	EOR n,X	2	55	+ +
A-A EOR M	Apsolutno	EOR nn	3	4D	+ +
	Apsolutno,X	EOR nn,X	3	5D	+ +
	Apsolutno,Y	EOR nn,Y	3	59	+ +
	Preindeksirano	EOR (n,X)	2	41	+ +
	Postindeksirano	EOR (n),Y	2	51	+ +
INC					NV-BDIZC
Povacanja sadr.	Nulta strana	INC n	2	E6	+ +
mamorija za jedan	Nulta strana,X	INC n,X	2	F6	+ +
M-M+1	Apsolutno	INC nn	3	EE	+ +
	Apsolutno,X	INC nn,X	3	FE	+ +
INX					NV-BDIZC
Poviscanja sadržaja rag. X za jadan (X-X+1)	Implicitno	INX	1	E8	+ +
INY					NV-BDIZC
Povećanje sadržaja rag. Y za jadan (Y-Y+1)	Implicitno	INY	1	C8	+ +

JMP					NV-BDIZC
Skok na memorijsku lokaciju	Apsolutno	JMP nn	3	4C	
	Indirektno	JMP (nn)	3	6C	
JSR					NV-BDIZC
Odlazak na potprogram (povratna adr. na stek)	Apsolutno	JSR nn	3	20	
LDA					NV-BDIZC
Punjenje akumulatora sadržajem memorijske lokacije	Neposredno	LDA #n	2	A9	+ +
	Nulta strana	LDA n	2	A5	+ +
	Nulta strana,X	LDA n,X	2	B5	+ +
	Apsolutno	LDA nn	3	AD	+ +
	Apsolutno,X	LDA nn,X	3	BD	+ +
	Apsolutno,Y	LDA nn,Y	3	B9	+ +
	Preindeksirano	LDA (n,X)	2	A1	+ +
	Postindeksirano	LDA (n),Y	2	B1	+ +
LDX					NV-BDIZC
Punjenje registra X sadržajem memorijske lokacije	Neposredno	LDX #n	2	A2	+ +
	Nulta strana	LDX n	2	A6	+ +
	Nulta strana,Y	LDX n,Y	2	B6	+ +
	Apsolutno	LDX nn	3	AE	+ +
	Apsolutno,Y	LDX nn,Y	3	BE	+ +
LDY					NV-BDIZC
Punjenje registra Y sadržajem memorijske lokacije	Neposredno	LDY #n	2	A0	+ +
	Nulta strana	LDY n	2	A4	+ +
	Nulta strana,X	LDY n,X	2	B4	+ +
	Apsolutno	LDY nn	3	AC	+ +
	Apsolutno,X	LDY nn,X	3	BC	+ +
LSR					NV-BDIZC
Pomernje bita u bajtu udesno za jedno mesto (memorije ili reg. A)	Akumulatorsko	LSR A	1	4A	+ ++
	Nulta strana	LSR n	2	46	+ ++
	Nulta strana,X	LSR n,X	2	56	+ ++
	Apsolutno	LSR nn	3	4E	+ ++
0 → 76543210 → C	Apsolutno,X	LSR nn,X	3	4E	+ ++
NOP					NV-BDIZC
Dva mašinska ciklusa čekanja	Implicitno	NOP	1	EA	
ORA					NV-BDIZC
Logičko sabiranje (ILI) sadržaja akumulatora i memorijske lokacije A=A OR M	Neposredno	ORA #n	2	09	+ +
	Nulta strana	ORA n	2	05	+ +
	Nulta strana,X	ORA n,X	2	15	+ +
	Apsolutno	ORA nn	3	0D	+ +
	Apsolutno,X	ORA nn,X	3	1D	+ +
	Apsolutno,Y	ORA nn,Y	3	19	+ +
	Preindeksirano	ORA (n,X)	2	01	+ +
	Postindeksirano	ORA (n),Y	2	11	+ +
PHA					NV-BDIZC
Prebacivanje sadržaja registra A na stek	Implicitno	PHA	1	48	
PHP					NV-BDIZC
Prebacivanje sadržaja registra P na stek	Implicitno	PHP	1	08	

PLA					NV-BDIZC
Punjenje registra A sadržajem sa steka	Implicitno	PLA	1	68	+ +
PLP					NV-BDIZC
Punjenje registra P sadržajem sa steka	Implicitno	PLP	1	28	sa steka
ROL					NV-BDIZC
Rotacija bita u bajtu	Akumulatorsko	ROL A	1	2A	+ ++
ulevo za jedno mesto	Nulta strana	ROL n	2	26	+ ++
(memorije ili reg. A)	Nulta strana,X	ROL n,X	2	36	+ ++
	Apsolutno	ROL nn	3	2E	+ ++
[76543210]-[C]	Apsolutno,X	ROL nn,X	3	3E	+ ++
ROR					NV-BDIZC
Rotacija bita u bajtu	Akumulatorsko	ROR A	1	6A	+ ++
udesno za jedno mesto	Nulta strana	ROR n	2	66	+ ++
(memorije ili reg. A)	Nulta strana,X	ROR n,X	2	76	+ ++
	Apsolutno	ROR nn	3	6E	+ ++
[C]-[76543210]	Apsolutno,X	ROR nn,X	3	7E	+ ++
RTI					NV-BDIZC
Povratak iz programskog prekida	Implicitno	RTI	1	40	sa steka
RTS					NV-BDIZC
Povratak iz potprograma	Implicitno	RTS	1	60	
SBC					NV-BDIZC
Oduzimanje sadržaja	Neposredno	SBC #n	2	E9	++ ++
memorije i invertovanog	Nulta strana	SBC n	2	E5	++ ++
bita prenosa od	Nulta strana,X	SBC n,X	2	F5	++ ++
akumulatora	Apsolutno	SBC nn	3	ED	++ ++
A-A-M-C	Apsolutno,X	SBC nn,X	3	FD	++ ++
	Apsolutno,Y	SBC nn,Y	3	F9	++ ++
	Preindeksirano	SBC (n,X)	2	E1	++ ++
	Postindeksirano	SBC (n),Y	2	F1	++ ++
SEC					NV-BDIZC
Setovanje bita C C-1	Implicitno	SEC	1	38	1
SED					NV-BDIZC
Setovanje bita D D-1	Implicitno	SED	1	F8	1
SEI					NV-BDIZC
Setovanje bita I I-1	Implicitno	SEI	1	78	1
STA					NV-BDIZC
Punjenje memorijske	Nulta strana	STA n	2	85	
lokacije sadržajem	Nulta strana,X	STA n,X	2	95	
akumulatora	Apsolutno	STA nn	3	8D	
	Apsolutno,X	STA nn,X	3	9D	
	Apsolutno,Y	STA nn,Y	3	99	
	Preindeksirano	STA (n,X)	2	81	
	Postindeksir	STA (n),Y	2	91	

STX					NV-BDIZC
Punjenje memorijske	Nulta strana	STX n	2	86	
lokacije sadržajem	Nulta strana,Y	STX n,Y	2	96	
registra X	Apsolutno	STX nn	3	8E	
STY					NV-BDIZC
Punjenje memorijske	Nulta strana	STY n	2	84	
lokacije sadržajem	Nulta strana,X	STY n,X	2	94	
registra Y	Apsolutno	STY nn	3	8C	
TAX					NV-BDIZC
Premeštanje A u X	Implicitno	TAX	1	AA	+ +
TAY					NV-BDIZC
Premeštanje A u Y	Implicitno	TAY	1	A8	+ +
TSX					NV-BDIZC
Premeštanje SP u X	Implicitno	TSX	1	BA	+ +
TXA					NV-BDIZC
Premeštanje X u A	Implicitno	TXA	1	8A	+ +
TXS					NV-BDIZC
Premeštanje X u SP	Implicitno	TXS	1	9A	+ +
TYA					NV-BDIZC
Premeštanje Y u A	Implicitno	TYA	1	98	+ +

7.8 PRIMERI PROGRAMIRANJA NA MAŠINSKOM JEZIKU

U prethodnim poglavljima izložena je materija neophodna za razumevanje i pisanje mašinskih programa. U ovom poglavlju biće prikazano nekoliko mašinskih programa u cilju ilustrovanja korišćenja mašinskih naredbi. Takođe je prikazan način realizacije mašinskih programa.

Mašinski programi se mogu unositi u računar na više načina. Najelementarniji, ali spor i težak način je unošenje kodova operacija i podataka pomoću bejzik naredbe POKE. Upotrebom bejzik programa za unošenje kodova, proces se može ubrzati. U primerima koji slede date su adrese memorijskih lokacija i kodovi koje treba uneti. Mogu se uneti pomoću datog programa HEX punjača.

```
10 ADR=49152:PRINTCHR$(147)
20 W$="":INPUT "KOD";H$:IF H$="S"THEN STOP
30 IF LEN(H$)<>2 GOTO 20
40 KOD=16*(ASC(H$)-48+7*(ASC(H$)>64))
50 KOD=KOD+ASC(RIGHT$(H$,1))-48+7*(ASC(RIGHT$(H$,1))>64)
60 POKE ADR,KOD
70 A3%=ADR/4096:A2%=(ADR-4096*A3%)/256
80 A1%=(ADR-4096*A3%-256*A2%)/16
90 A0=ADR-4096*A3%-256*A2%-16*A1%
100 A=A3%:GOSUB 200:A=A2%:GOSUB 200:A=A1%:GOSUB 200:A=A0:GOSUB 200
110 PRINT"]",W$,H$:ADR=ADR+1:GOTO20
200 A=A+48-7*(A>=10):W$=W$+CHR$(A):RETURN
```

Programskom linijom 10 određeno je da se unošenje obavlja od memorijske lokacije sa adresom 49152. Svi primeri dati u ovom poglavlju se unose od te lokacije. Unošenje hek-

sadecimalnih vrednosti se obavlja u rastućem poretku, od nižih ka višim adresama, što znači da kodove pri unošenju treba uzimati iz primera sa leva na desno i odozgo na dole. Taj redosled je najbolje ilustrovan u primeru 1.

Nakon unošenja dve heksadecimalne cifre, pritiskom na taster RETURN vrši se upisivanje njihovog koda u memorijsku lokaciju. Po unošenju svih kodova pritiskom na taster S i RETURN ostvaruje se završetak punjenja.

Najefikasniji način upisivanja mašinskih programa je korišćenje programa asemblera. Upisivanje se izvršava unošenjem mnemoničkih oznaka naredbi, a program asembler generiše odgovarajuće kodove koje upisuje na željene adrese. Primeri mašinskih programa dati u knjizi su uneti korišćenjem programa asemblera PROFI – ASS 64, V2.0. U primeru br. 1 prikazane su osnove korišćenja tog programa.

Za smeštanje mašinskih programa može se u principu upotrebiti bilo koji deo memorije. Praktično to nije moguće izvesti jer bi došlo do preklapanja unetog mašinskog programa sa programom ili podacima neophodnim za regularni rad bejzik interpretera i operativnog sistema.

Memorijski prostor od adrese 49152 (videti organizaciju memorije) je najčešće korišćen prostor za smeštanje mašinskih programa. Razlog za to je što se taj prostor ne upotrebljava pri korišćenju Komodora u bejziku.

Mašinski programi se mogu smestiti i u prostor ispred onog koji zauzima bejzik program. U cilju formiranja tog prostora potrebno je premestiti početak bejzik programa na višu adresu. To se postiže unošenjem vrednosti nove adrese u sistemsku promenljivu TXTTAB (u lokaciju 43 niži bajt adrese, a u lokaciju 44 viši bajt adrese). Pri tome je obavezno da se u lokaciju koja prethodi onoj na koju ukazuje promenljiva TXTTAB, unese vrednost nula. Takođe treba izvršiti naredbu **NEW** pre i posle unošenja nove vrednosti u promenljivu.

Mašinski programi se mogu smestiti i iza bejzik programa. U tom slučaju potrebno je spustiti kraj prostora za bejzik programe. To se postiže unošenjem vrednosti nove adrese kraja u sistemsku promenljivu MEMSIZ (u lokaciju 55 niži bejt adrese, a u lokaciju 56 viši bajt adrese). Pre i posle unošenja potrebno je izvršiti naredbu **CLR**. Moguće je koristiti RAM memoriju na istim lokacijama na kojima je ROM memorija operativnog sistema i bejzik interpretera. Tada je potrebno obavljati preklapanje memorija, ali samo za naredbe očitavanja iz RAM memorije. Taj prostor je pogodan, ne kao radna memorija, već kao prostor za smeštanje veće količine podataka.

Startovanje unetog mašinskog koda (programa) ostvaruje se naredbom **SYS** nn, gde je nn adresa od koje se želi izvršavanje mašinskog programa. Ta naredba sadržaj programskog brojača (PC) postavlja na datu vrednost nn. Time je postignuto da procesor izvršava naredbe od adrese nn. Prethodna vrednost programskog brojača se čuva na steku i odatle se uzima na kraju mašinskog programa kao povratna adresa u bejzik program. To je po pravilu ostvareno naredbom RTS na kraju mašinskog programa.

Primer 1

Navedeni primer mašinskog programa ilustruje upotrebu naredbi premeštanja podataka. Opisan je i način unošenja programa pomoću napred izloženog programa za unošenje kodova. Takođe je prikazan način upotrebe programa asemblera za unošenje programa.

Program je dužine 9 bajta. Početak programa je na memorijskoj lokaciji $C000 (49152), a završetak na lokaciji $C008. U sledeće tri kolone su date adrese lokacija, kodovi i mnemonika mašinskih naredbi.

adresa	kod	mnemonika
C000	A9 01	LDA #1
C000	85 02	STA 2
C004	AA	TAX
C005	8E CC 05	STX $05CC
C008	60	RTS

Prva naredba je naredba neposrednog punjenja akumulatora. U primeru naredba puni akumulator brojnom vrednošću 1. Naredba je dužine dva bajta i unosi se u memorijske lokacije $C000 i $C001. Druga naredba je naredba kojom se sadržaj akumulatora smešta u memorijsku lokaciju broj 2. Pri tome sadržaj akumulatora ostaje nepromenjen. Naredba je takođe dvobajtna. Prvi bajt je kôd operacije, a drugi je adresa lokacije, koja se nalazi na nultoj strani. Izvršenjem prve dve naredbe brojna vrednost 1 je smeštena u memorijsku lokaciju 2. To se može proveriti naredbom PRINT PEEK (2). S obzirom da ne postoje naredbe neposrednog punjenja memorijskih lokacija, u primeru je upotrebljen akumulator.

Treća naredba u programu je naredba prebacivanja sadržaja akumulatora u registar X. Naredba je jednobajtna i nalazi se na lokaciji $C004. Četvrta naredba sadržaj registra X premešta u memorijsku lokaciju 1464 ($05CC). Pri tome je sadržaj registra X ostao nepromenjen. Naredba je trobajtna. Prvi bajt je kôd operacije, a drugi i treći su kodovi adrese (prvo bajt manje težine). U ovoj naredbi način adresiranja je apsolutan.

Brojna vrednost 1 je trećom naredbom upisana u registar X, a četvrtom u lokaciju 1484. Ta lokacija se nalazi u ekranskoj memoriji i odgovara karakteru na sredini ekrana. Izvršenjem naredbi na sredini ekrana će biti napisano slovo A (ekranski kod slova A je 1). Poznavanje organizacije memorije je neophodno za razumevanje ovoga, a i većeg dela narednih primera.

Promenom sadržaja akumulatora menja se i karakter koji se ispisuje, a promenom adrese u četvrtoj naredbi menja se i mesto ispisivanja karaktera.

Poslednja naredba je obavezna naredba povratka iz mašinskog programa.

Unošenje programa u računar se ostvaruje unošenjem njegovih kodova. Za to se može koristiti dati program HEX punjač, na sledeći način. Po startovanju programa upisuje se kôd A9 i pritiska taster RETURN. Zatim se opisuje kôd 01 uz ponovni pritisak tastera RETURN. Postupak unošenja se nastavlja narednim kodovima. U ovom primeru kodovi koji se dalje unose su: 85. 02, AA, 8E, CC, 05 i 60. Završetak unošenja se ostvaruje pritiskom tastera S i RETURN.

Startovanje unetog mašinskog programa se ostvaruje naredbom **SYS** 49152. U memorijsku lokaciju 2 će biti upisana vrednost 1, a u lokaciju 1484 takođe vrednost 1, čime će biti ispisano slovo A na sredini ekrana (poželjno je prethodno obrisati ekran).

Kodovi programa moraju biti tačno uneti. U protivnom skoro je sigurno da program neće funkcionisati i da neće biti moguć povratak iz mašinskog programa. U nekim slučajevima uspešan povratak se može obaviti pritiskom na tastere RUN/STOP i RESTORE.

Za efikasnije unošenje kodova mogu se upotrebiti programi asembleri. Na ovom mestu će biti opisani osnovni pojmovi potrebni za korišćenje programa asemblera PROFI — ASS 64, V2.0.

Po učitavanju i startovanju programa asemblera može se pristupiti ispisivanju programskih linija. Slično programu u bejziku, linija se sastoji od broja linije i naredbi u liniji. U ovom slučaju naredbe su mašinske. Osim mašinskih naredbi mogu se upisivati i komande asemblera. To su pseudo naredbe koje određuju rad asemilera.

Prikazan je kompletan asemblerski listing programa iz ovog primera.

```
10 SYS 8*4096
20 .OPT,OO,P
30 *= $C000
40 :         LDA #1
50 :         STA 2
60 :         TAX
70 :         STX 1484
80 :         RTS
90 .END
```

Linijom 10 se startuje prevođenje navedenih naredbi u kodove i smeštanje kodova od naznačene adrese. Linija 20 određuje smeštanje kodova i ispisivanje za vreme prevođenja. Linija 30 određuje adresu lokacije od koje će biti smešteni kodovi (prazno mesto iza znaka jednakosti je obavezno). Od linije 40 do linije 80 su mašinske naredbe. Dvotačke su stavljenje da bi omogućile pregledniji ispis naredbi. Linija 90 označava kraj programa napisanog pomoću asemblera.

Praktično prve tri linije, i poslednja linija su obavezne u svakom korišćenju asemblera. One startuju, određuju i završavaju rad asemblera.

Nakon ispisivanja svih linija, treba izvršiti naredbu **RUN** čime će se aktivirati asembler. Pri tome će se dobiti novi ispis na ekranu. On sadrži heksadecimalne adrese lokacija i kodove kojim su te lokacije napunjene. Takođe će dati i zveštaje o greškama. Po uklanjanju grešaka ispisivanja i po dobijanju izveštaja da nema grešaka (NO ERRORS) u naznačenim lokacijama je formiran ispravan mašinski kôd. Startovanje dobijenog mašinskog programa se može ostvariti naredbom SYS 49152, kako je napred opisano.

Na ovom mestu je dat samo neophodan minimum za korišćenje asemblera. Neke njegove mogućnosti su prikazane u narednim primerima. Potpuno upoznavanje sa asemblerom je neophodno za uspešno programiranje na mašinskom jeziku.

U sledećim primerima mašinskih programa biće dati asemblerski listinzi programa dobijeni po startovanju asemblera. S obzirom da oni sadrže kodove, unošenje programa se može obavljati i programom HEX punjač.

Primer 2

U ovom primeru su prikazane naredbe punjenja registara sadržajem memorijske lokacije i naredbe stavljanja i očitavanja podataka sa steka.

adresa	kod	mnemonika
C000	AD 20 D0	LDA $D020
C003	48	PHA
C004	AD 21 D0	LDA $D021
C007	8D 20 D0	STA $D020
C00A	68	PLA
C00B	8D 21 D0	STA $D021
C00E	60	RTS

Program je dužine 15 bajta. Prva tri su kodovi za punjenje registra A sadržajem memorijske lokacije $D020 (53280). Sadržaj te lokacije određuje boju obodnog dela ekrana. Druga naredba je jednobajtna. To je naredba koja sadržaj akumulatora prebacuje na stek (memorijski prostor za privremeno smeštanje podataka). Treća naredba programa puni akumulator sadržajem memorijske lokacije $D021 (53281). Sadržaj te lokacije određuje boju središnjeg dela ekrana. Četvrtom naredbom je taj sadržaj smešten u lokaclju koja određuje

boju obodnog dela ekrana. Time je obodni deo ekrana dobio boju središnjeg dela. Peta naredba je naredba steka. Ona akumulatoru dodeljuje vrednost koja je poslednja stavljena na stek. To je prethodno smeštena vrednost boje obodnog dela ekrana. Šesta naredba tu vrednost iz akumulatora preslikava u memorijsku lokaciju za određivanje boje središnjeg dela ekrana. Tako je ostvareno da središnji i obodni deo ekrana zamene boje. Šesta naredba je naredba povratka iz mašinskog programa.

Zamena vrednosti dve memorijske lokacije može se ostvariti i bez korišćenja steka već upotrebom treće, pomoćne lokacije. Upotreba steka je u ovom slučaju opravdanija. Osim što je tako program kraći i brži, stek se koristi za ono za šta je i namenjen, tj. za privremeno smeštanje podataka.

Na steku se istovremeno može nalaziti više podataka. Pri tome je značajno da se sa steka, naredbama steka, mogu očitavati podaci po redosledu obrnutom od redosleda stavljanja na stek. To znači da se zadnje stavljen podatak očitava prvi. U radu sa stekom značajno je takođe da broj naredbi stavljanja podataka na stek i broj naredbi vraćanja podataka sa steka, u programu bude jednak. Razlog je u korišćenju steka za čuvanje povratnih adresa.

Izloženi program se startuje, kao i svi primeri u ovom delu knjige, naredbom SYS 49152.

Dat je odgovarajući asemblerski listing programa u ovom primeru.

```
10 SYS 8*4096
20 .OPT,OO,P
30 :M1 = $0020
40 :M2 = $D021
50 *= $C000
100 :         LOA M1
110 :         PHA
120 :         LOA M2
130 :         STA M1
140 :         PLA
150 :         STA M2
160 :         RTS
170 .ENO
```

U linijama 30 i 40 je iskorišćena mogućnost asemblera da radi sa simboličkim oznakama, labelama. Labelama M1 i M2 dodeljene su vrednosti adresa koje se koriste u programu (pri tome su prazna polja ispred i iza znakova jednakosti obavezna).

Primer 3

U ovom primeru biće prikazane artmetičke naredbe. Dati program prikazuje naredbe sabiranja i oduzimanja sadržaja akumulatora i memorijske lokacije.

adresa	kod	mnemonika
C000	EA	NOP
C001	A9 nn	LOA #nn
C003	69 mm	AOC #mm
C005	85 02	STA 2
C007	60	RTS

Navedeni program demonstrira naredbu **ADC.** Umesto nn i mm treba uneti vrednosti nad kojima se želi izvršiti operacija sabiranja. Te vrednosti se smeštaju u lokacije sa adresama $C002 (49154) i $C004 (49156).

U prvom bajtu programa nalazi se kôd $EA koji odgovara naredbi kontrole procesora. Nailaskom na naredbu **NOP**, procesor jednostavno prelazi na izvršavanje naredne naredbe. **NOP** naredba je tu postavljena da bi se bejzik naredbom POKE 49152,56 zamenila sa naredbom **SEC** koja postavlja bit prenosa na jedinicu. Time se može demonstrirati osmobitno sabiranje sa prenosom.

Isti program zamenom koda na adresi 49155 (naredba **ADC**), kodom naredbe **SBC** ($E9 = 233) prikazuje osmobitno oduzimanje. Zamenom koda na adresi 49152, kodom naredbe za postavljanje bita prenosa na nulu (kôd za **CLC** je $18 = 24) ili za postavljanje na jedinicu, može se prikazati uticaj bita C na rezultat.

U datom primeru operacija sabiranja tj. oduzimanja se obavlja između akumulatora i neposrednog brojnog podatka. Razni načini adresiranja omogućuju pristup podatku koji se nalazi bilo gde u memoriji.

Četvrtom naredbom u programu rezultat se smešta u memorijsku lokaciju sa adresom 2. To je lokacija na nultoj strani koja nije korišćena od strane Komodora u njegovom normalnom radu. Po izvršenju programa naredbom **SYS 49152** rezultat operacije se može proveriti naredbom **PRINT PEEK(2).**

Aritmetičke naredbe povećanja i smanjenja za jedan sadržaja registara X i Y i memorijskih lokacija su ostavljene čitaocu da ih sam upotrebi. Svakako da će u primerima koji slede biti upotrebljene i ove naredbe.

Primer 4

Ovaj primer prikazuje naredbe poređenja, uslovnog grananja, povećanja i smanjenja za jedan.

```
adresa   kod          menmonika
C000     A2 00           LDX #0
C002     A9 41           LDA #$41
C004     C9 41           CMP #$41
C006     F0 04           BEQ L1
C008     CA              DEX
C009     86 02           STX 2
C00B     60              RTS
C00C     E8           L1 INX
C00D     86 02           STX 2
C00F     60              RTS
```

U prvoj naredbi se registar X puni vrednošću 0. U drugoj naredbi se registar A puni vrednošću $41 (65). Treća naredba je naredba poređenja sadržaja akumulatora sa vrednošću $41. Kako je već rečeno, naredba poređenja je po efektu slična naredbi oduzimanja, s tim što se sadržaj akumulatora ne menja. Indikatori stanja se postavljaju isto kao i u slučaju oduzimanja. Za vrednosti navedene u primeru, rezultat oduzimanja je nula, što će postaviti indikator nule (Z bit) na jedinicu.

Četvrta naredba je naredba uslovnog grananja u slučaju da je indikator nule jednak jedinici. S obzirom da je taj uslov ispunjen, program će nastaviti sa izvršavanjem od memorijske lokacije sa adresom za 4 većom nego u slučaju da uslov nije bio ispunjen. To je lokacija $C00C na kojoj se nalazi naredba koja će povećati sadržaja registra X za **1 (INX).** To će biti potvrđeno naredbom **PRINT PEEK(2),** po izvršenju programa, kojom će se dobiti rezultat broj 1.

Ukoliko se akumulator, u drugoj naredbi, napuni nekom drugom vrednošću, neće doći do ispunjenja uslova potrebnog za grananje. U tom slučaju će se nastaviti sa izvršavanjem od naredbe **DEX**, što će u memorijskoj lokaciji 2 rezultovati brojem 255.

U osmoj naredbi, **INX**, memorijskoj lokaclji $C00C pridodato je ime, labela L1. Četvrta naredba se poziva na labelu L1, odnosno na njenu memorijsku lokaciju. Označavanje labela, iako ne predstavlja standardnu mnemoniku procesora 6510, nezamenljiv je deo asemblerskog programa.

Na osnovu ovog primera i svega što je do sada izloženo mogu se bez većih teškoća izvršiti izmene koje omogućuju prikazivanje ostalih naredbi grananja.

Primer 5

Program u ovom primeru prikazuje logičke operacije množenja, sabiranja i isključivog sabiranja. Naredbe koje omogućuju te operacije su **AND, ORA** i **EOR.**

adresa	kod	menmonika
C000	A9 nn	LDA #nn
C002	8D 30 C0	STA 49200
C005	A9 mm	LDA #mm
C007	2D 30 C0	AND 49200
C00A	8D 30 C0	STA 49200
C00D	60	RTS

Umesto nn i mm treba uneti 8-bitne vrednosti nad kojima se želi izvršavanje logičkog množenja. Izvršenjem programa **(SYS 49152)** rezultat će biti smešten u memorijsku lokaciju 49200 ($C030).

Kao što je i ranije rečeno, za razumevanje logičkih operacija neophodno je predstavljanje brojeva, koji učestvuju u operacijama, u binarnom obliku. Mala vežba iz programiranja u bejziku bi se sastojala u pisanju programa za pretvaranje decimalnih brojeva u binarni oblik. Primer 8 prikazuje moguće rešenje na mašinskom jeziku.

Zamenom kôda naredbe **AND** kodom naredbe **ORA,** odnosno **EOR**, omogućuje se izvršavanje programa i dobijanje rezultata za naredbe logičkog sabiranja i logičkog isključivog sabiranja.

Primer 6

U ovom primeru je demonstrirana upotreba naredbi iz grupe za pomeranje bita.

adresa	kod	mnemonika	komentar
C000	A2 00	LDX #0	Sadržaj X reg. postaviti na nulu
C002	A0 nn nn	LDA nnnn	nnnn-adresa željanog bajta
C005	4A	LSR A	Pomaranja bita u desno
C006	90 03	BCC L1	Bit 0, da li ja nula (paran broj)
C008	86 02	STX 2	Nija. U lokaciju 2 smašta sa nula
C00A	60	RTS	Povratak
C00B	E8	L1 INX	Jasta. Povaćanja X za jadan
C00C	86 02	STX 2	U lokaciju 2 smašta se jadinica
C00E	60	RTS	Povratak

Program na osnovu vrednosti bita najmanje težine (bit 0) u adresiranoj memorijskoj lokaciji, utvrđuje da li je vrednost njenog sadržaja paran ili neparan broj. Vrednost bita 0 je očitana njegovim premeštanjem u indikator prenosa (bit C).

Program se izvršava uobičajenom naredbom **SYS 49152**, a rezultat se smešta u lokaciju sa adresom 2. Za parni broj rezultat će biti 1, a za neparni 0.

Primer 7

Naredbe odlaska na izvršavanje potprograma i povratka su prikazane u ovom primeru.

```
adresa    kod              mnemonika     komentar

C000      A2 01               LDX #1     Sadržaj X neka buda 1
C002      20 0B C0            JSR L1     Idi na potprogram L1
C005      20 0B C0            JSR L1     Idi ponovo na L1
C008      86 02               STX 2      Sadržaj X u lokac. 2
C00A      60                  RTS        Povratak
C00B      E8               L1 INX        Potprogram. Povaćaj X
C00C      60                  RTS        Povratak u glavni program
```

U glavnom programu se potprogram poziva dva puta i svaki put se sadržaj registra X povećava za jedan. Rezultat, broj 3, će biti smešten u memorijsku lokaciju 2.

Za povratak iz mašinskog potprograma služi ista naredba (RTS) kao i naredba za povratak iz mašinskog programa.

Primer 8

Ovaj primer prikazuje upotrebu naredbi rotacije. Takođe prikazuje formiranje programske petlje u mašinskom programu.

```
adrasa    kod              mnemonika     komentar

C000      A9 nn               LDA #nn      nn-željena vrednost
C002      A0 08            L1 LDY #8       Za ponavljanje 8 puta
C004      A2 30               LDX #$30     $30-kod cifre 0
C006      6A                  ROR A        Za bit udesno sadržaj reg. A
C007      48                  PHA          Sacuvaj privremeno sarzaj A
C008      90 01               BCC L2       Ako nema prenosa skok na L2
C00A      E8                  INX          Ima. $31-kod cifre 1
C00B      8A               L2 TXA          Nema. Kôd cifre u reg. A
C00C      99 46 04            STA 1094,Y   Iz njega u video memoriju
C00F      68                  PLA          Vrati sačuvanu vrednost A
C010      88                  DEY          Smanji brojač petlja
C011      D0 F1               BNE L1       Ako nija 0 nastavak od L1
C013      60                  RTS          Jeste. Povratak
```

Program ispisuje sadržaj registra A u binarnom obliku. Četvrtom naredbom se obavlja rotacija sadržaja A registra za jedno mesto u desno. Krajnje desni bit rotacijom prelazi u bit prenosa definišući uslov za izvođenje naredbe uslovnog grananja. U slučaju da je bit prenosa postavljen na jedinicu, neće doći do skoka i vrednost sadržaja registra X će se povećati za jedan. Novi sadržaj registra X je $31, što je kod broja 1. Na ekranu će se ispisati 1. U slučaju da je ispunjen uslov grananja preskače se naredba povećanja sadržaja registra X i na ekranu se ispisuje 0. Postupak se ponavlja 8 puta ostvarujući ispisivanje osmobitne vrednosti u binarnom obliku.

Mesto ispisivanja je u gornjem desnom uglu ekrana i određeno je vrednošću 1094 ($0446). Ispisivanje cifara na susednim mestima je ostvareno adresiranjem indeksnim registrom Y. Smanjivanjem registra Y u svakom prolazu kroz petlju određuje se nova memorijska lokacija u video memoriji.

Izvršenje programa se ostvaruje naredbom **SYS 49152**, a pre toga je poželjno obrisati sadržaj ekrana.

Primer 9

Programi u prethodnim primerima su prikazali upotrebu određenih mašinskih naredbi. Program naveden u ovom primeru ima za cilj da potvrdi prethodno izloženo i da upozna čitaoca sa problematikom koja se javlja pri formiranju namenskih mašinskih programa.

adresa	kod	mnemonika	komentar
C000	A9 00	LDA #0	Ekranski kod prvog karaktera (@)
C002	85 04	STA 4	Sačuvati npr. u lokaciji 4
C004	A9 00	L3 LDA #0	U lokacije 2 i 3 staviti vrednost 1024
C006	85 02	STA 2	U lokaciju 2 vrednost 0 (niži bajt)
C008	A9 04	LDA #4	U lokaciju 3 vrednost 4 (viši bajt)
C00A	85 03	STA 3	(1024=1*0+256*4)
C00C	A0 00	LDY #0	Priprema brojača petlje i lokac. ekrana
C00E	A5 04	LDA 4	Ucitavanje sacuvanog ekranskog koda
C010	91 02	L1 STA (2),Y	Smeštanje u lokac. ekranske memorije
C012	C8	INY	Ka sledećoj lokaciji
C013	D0 02	BNE L2	Ako nije 256-a (nulta) grananje na L2
C015	E6 03	INC 3	Ako jeste povećanje višeg bajta za 1
C017	C0 E8	L2 CPY #232	Da li je 232-a lok.? (1000=3*256+232)
C019	D0 F5	BNE L1	Ako nije postupak se ponavlja od L1
C01B	A2 07	LDX #7	Jeste. Da nije i poslednja?
C01D	E4 03	CPX 3	Ako jeste viši bajt je 7
C01F	D0 EF	BNE L1	Nije poslednja. Nastavak od L1
C021	E6 04	INC 4	Jeste (ceo ekran). Sledeći karakter
C023	A5 04	LDA 4	Smešta se u registar A
C025	C9 80	CMP #128	Da li je prošlo svih 128 karaktera?
C027	D0 DB	BNE L3	Nije. Popunjavanje ekrana sledećim kar.
C029	60	RTS	Jeste. Povratak

Dati program popunjava ceo ekran karakterom (1000 karaktera) čiji je ekranski kod 0 (@), zatim karakterom sa kodom 1 (A), pa sa kodom 2 (B), itd. po rastućim ekranskim kodovima (videti tabelu ekranskih kodova u dodatku). Ukupno 128 različitih sadržaja ekrana.

Ne može se reći da program ima veću praktičnu primenu, ali njegovo rešavanje dobro prikazuje problematiku programiranja na mašinskom jeziku. Takođe atraktivno demonstrira brzinu programa napisanog na mašinskom jeziku.

Za razumevanje ovoga programa preporučuje se čitaocu da obavezno napravi dijagram toka na osnovu datih komentara. Potrebno je takođe dati sledeća objašnjenja: Memorija za dobijanje slike tekstualnog ekrana počinje od lokacije 1024 (256*4), a završava se na lokaciji 2023 (231 + 256*7). Za adresiranje lokacija ekranske memorije, i za upisivanje kodova karaktera u njih, upotrebljeno je indirektno adresiranje postindeksirano registrom Y. Razlog zašto nije upotrebljeno apsolutno indeksirano adresiranje, kao što je učinjeno u prethodnom primeru, je u tome što je deo memorije u koji se vrši upisivanje, tj. video memorija, veća od 255 bajta. Na ovom mestu se može reći da se program može realizovati i upotrebom naredbe **STA** sa apsolutnim indeksiranim adresiranjem, ali bi to zahtevalo promenu podataka tj. naredbi programa u toku izvršavanja, od strane samog programa. Iako se takvim načinom pisanja programa mogu dobiti brža i kraća rešenja, takav postupak se ne može preporučiti niti nazvati dobrim programiranjem.

Čitaocu se predlaže da proširi dati program uvođenjem željenog usporenja izvršavanja programa.

Startovanje programa se obavlja uobičajenom naredbom **SYS 49152.**

Primer 10

Poslednji primer u ovom poglavlju prikazuje korišćenje ROM rutina u formiranju mašinskih programa. Primer ima za cilj da uputi čitaoca na upoznavanje organizacije memorije i korišćenje ROM rutina. Primer takođe ilustruje naredbe programskog prekida i kontrolu procesora.

```
adresa   kod          mnemonika       komentar

C000     A9 52        LDA #"R"        Punjenje lokacija od adrese $C100
C002     8D 00 C1     STA $C100       kodovima karaktera R U N
C005     A9 55        LDA #"U"
C007     8D 01 C1     STA $C101
C00A     A9 4E        LDA #"N"
C00C     8D 02 C1     STA $C102
C00F     A9 0D        LDA #13         Kod za RETURN
C011     8D 03 C1     STA $C103
C014     A9 00        LDA #0          Obavezna nula za kraj stringa
C016     8D 04 C1     STA $C104
C019     78           SEI             Prekid mora biti onemogućen za vreme
C01A     A9 26        LDA #$26        promene vrednosti vektora prekida
C01C     8D 14 03     STA $314        na lokacijama $314 i $315
C01F     A9 C0        LDA #$C0        rutina za obradu prekida je
C021     8D 15 03     STA $315        na adresi $C026
C024     58           CLI             omogućenje prekida
C025     60           RTS             Povratak
C026     20 9F FF     JSR $FF9F       Odlazak na rutinu očitavanja tastature
C029     A6 C6        LDX $C6         X jednako dužini stringa u baferu
C02B     F0 16        BEQ L1          Ako je dužina stringa nula povratak
C02D     CA           DEX             X ukazuje na prethodno uneti karakter
C02E     BD 77 02     LDA $277,X      A jednako kodu tog karaktera
C031     C9 88        CMP #$88        Da li je to f7?
C033     D0 0E        BNE L1          Ako nije onda povratak
C035     A0 FF        LDY #$FF        Punjenje bafera sa RUN i RETURN
C037     E8        L2 INX
C038     C8           INY
C039     B9 00 C1     LDA $C100,Y
C03C     9D 77 02     STA $277,X
C03F     D0 F6        BNE L2
C041     86 C6        STX $C6         X ukazuje na novu dužinu str. u baferu
C043     4C 31 EA L1  JMP $EA31       Povratak preko za to predvidjene rutine
```

Program zadaje funkcijskom tasteru f7 string "RUN" + CHR$(13). Time je postignuto da se pritiskom jednog tastera izvršava program koji se nalazi u računaru.

Program se sastoji iz dva dela. U prvom delu se od memorijske lokacije $C100 smešta potreban niz kodova za naredbu RUN. Takođe se obavlja promena vrednosti vektora prekida (lokacije $314 i $315). Nova vrednost ukazuje na drugi deo programa, na rutinu za obradu prekida. Taj deo programa će se izvršavati 50 puta u sekundi, nailaskom signala IRQ na odgovarajući izvod mikroprocesora.

U drugom delu programa 50 puta u sekundi se odlazi na izvršavanje ROM rutine koja očitava tastaturu. Kodovi pritisnutih tastera će biti smešteni u bafer (prihvatnu memoriju)

tastature koji počinje od lokacije $277. Broj elemenata u baferu je određen sadržajem lokacije $C6.

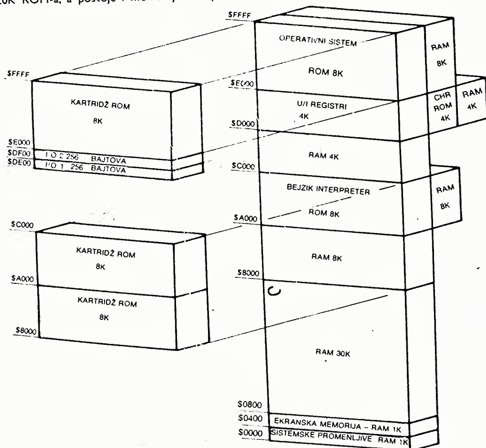
Ako je utvrđen pritisak na taster f7, u bafer se ubacuju kodovi za **RUN** i 〈RETURN〉. Bejzik interpreter će očitavanjem sadržaja bafera startovati bejzik program koji se nalazi u memoriji.

Programski prekidi procesora pružaju velike mogućnosti u rešavanju raznolikih problema. Na programeru je da pravilno odgovori na pitanja o pravdanosti njihove široke primene.

8
Organizacija memorije i upotreba ROM rutina

8.1 ORGANIZACIJA MEMORIJE

Mikroprocesor 6510 ima šesnaestobitnu adresnu magistralu preko koje može da adresira maksimalno 64Kbajta memorije. U Komodoru se međutim nalazi 64K RAM-a i 20K ROM-a, a postoje i memorijski mapirani registri ulazno/izlaznih jedinica. Takođe se



Sl. 8.1. Organizacija memorije

mogu dodati i spoljašnji ROM moduli – kartridži. Cela memorija nije aktivna istovremeno već se njeni pojedini segmenti uključuju i isključuju zavisno od potreba mikroprocesora i video kontrolera. Upravljanje memorijom obavlja poseban hardverski sklop (engl. memory management unit). Za detalje o njegovom funkcionisanju pogledati poglavlje 11. Hardver.

Sa slike 8.1. se može videti da se na lokacijama $0000 do $8000 nalazi RAM koji je dostupan mikroprocesoru, a oblast $0400 do $0800 je dostupna i video kontroleru.

Segment $8000 – $A000 čini RAM ali ukoliko je priključen kartridž pri čemu je linija $\overline{EXROM}$=0 (videti poglavlje 12. Konstrukcije) RAM se isključuje i na njegovo mesto se postavlja ROM.

Segment $A000 – $C000 čini ROM u kome je bejzik interpreter. Iza njega (na istim adresama) nalazi se RAM kome u ovoj konfiguraciji može da pristupi samo video kontroler. Ukoliko je linija $\overline{LORAM}$=0, uključuje se RAM, a isključuje bejzik ROM. U slučaju da je prisutan kartridž, pri čemu je linija $\overline{GAME}$=0, RAM se isključuje.

Segment $C000 – $D000 čini RAM.

Segment $D000 – $E000 ima najviše slojeva. Standardno se tu nalaze memorijski mapirani registri ulazno izlaznih jedinica. Iza se nalazi karakter ROM kome može da pristupi samo video kontroler, a iza karakter ROM-a je 4K RAM. U oblasti $D800 – DBFF nalazi se i kolor RAM kome pristupaju mikroprocesor i video kontroler. Oblast $DE00 – E000 mogu koristiti spoljašnje ulazno izlazne jedinice.

Segment $E000 – FFFF čini ROM operativnog sistema (Kernal) koji se može isključiti linijom $\overline{HIRAM}$=0. Tada na njegovo mesto dolazi RAM. Ukoliko je u istom trenutku prisutan i kartridž, pri čemu je linija $\overline{GAME}$=0, uključuje se njegov ROM, a RAM se isključuje.

8.2 SISTEMSKE PROMENLJIVE

Sistemske promenljive su sadržaji memorijskih lokacija od 0 do $400 (1024). To su pomoćne veličine koje postavljaju i koriste operativni sistem i bejzik interpreter. Pogodnom upotrebom i menjanjem sistemskih promenljivih, korisnik može znatno proširiti mogućnosti računara i pojednostaviti programiranje. Sistemske promenljive se mogu podeliti u nekoliko grupa:

Status – U ovu grupu spadaju promenljive koje ukazuju na određeno stanje računara u toku rada. Takođe, računar na osnovu njihovih vrednosti određuje kojim, od nekoliko različitih puteva u programu treba da krene.

Pokazivači – To su promenljive koje se koriste pri indirektnom adresiranju. One sadrže pravu adresu podatka i to tako da adresirana lokacija sadrži niži, a sledeća lokacija viši bajt prave adrese. Pokazivači se obično nalaze na nultoj strani memorije ($00 – $FF).

Vektori – To su promenljive koje sadrže adresu bezuslovnog skoka. Njihovim sadržajem se puni programski brojač, a zatim počinje izvršavanje programa od te nove adrese.

Tabele – To su grupe sistemskih promenljivih iste namene.

Baferi – Delovi memorije predviđeni za privremeno čuvanje podataka u periodu dok ne budu obrađeni. Ukoliko su to pojedini bajtovi, mogu se smatrati registrima.

Brojači – Sistemske promenljive čiji se sadržaj menja pri svakom koraku nekog brojanja.

Interno – Određene memorijske lokacije se koriste za namene koje je teško svrstati u neku od prethodnih grupa. U pitanju su često neke interne operacije ili neiskorišćene lokacije.

U nastavku je dat spisak svih sistemskih promenljivih po rastućim adresama. Prvo je data adresa u heksadecimalnom, a u zagradi je u decimalnom obliku. Zatim sledi ime, a iza zareza broj lokacija (bajtova) koje zauzima, kao i tip promenljive: S — status, P — pokazivač, V — vektor, T — tabela, R — bafer, B — brojač i prazno za interne i slobodne lokacije.

S0 (0) – MPDDR, 1 R
Mikroprocesorski registar smera prenosa podataka preko internog ulazno/izlaznog registra.

S1 (1) – MPDAT, 1 R
Mikroprocesorski interni osmobitni ulazno/izlazni registar.

S2 (2) – XXXX, 1
Slobodna lokacija.

S3 (3) – ADRAY1, 2 V
Ovaj vektor pokazuje na rutinu koja konvertuje broj iz oblika sa pokretnim zarezom u celobrojnu vrednost sa predzankom.

S5 (5) – ADRAY2, 2 V
Ovaj vektor pokazuje na rutinu koja konvertuje ceo broj sa predznakom u oblik sa pokretnim zarezom.

S7 (7) – CHARAC, 1
Karakter koji se trenutno traži u bejzik tekstu prvo se smešta u ovu lokaciju.

S8 (8) – ENDCHR, 1
Karakter kojim se završava naredba ili znak navoda prvo se smešta ovde, a zatim se traži u bejzik tekstu.

S9 (9) – TRMPOS, 1
Trenutna pozicija kursora koju koriste funkcije TAB i SPC za računanje sledeće pozicije.

SA (10) – VERCK, 1 S
Bejzik koristi istu Kernal rutinu i za LOAD i za VERIFY s tim što se razlika pravi na osnovu sadržaja A registra pri ulasku u nju. Promenljiva VERCK se postavlja na 0 ako je u pitanju naredba LOAD, a na 1 ako se radi o VERIFY. Odgovarajuća vrednost se stavlja i u A registar pre poziva Kernal LOAD rutine.

SB (11) – COUNT, 1
Brojač tokena i karaktera koje učitava bejzik iz bafera S200 (512). Po učitavanju poslednjeg karaktera iz linije, COUNT sadrži dužinu te linije. Druga funkcija COUNT je držanje broja indeksa matrice sa kojom se radi.

SC (12) – DIMFLG, 1 S
Ovu lokaciju koriste rutine koje rezervišu prostor za matrice u memoriji. Na osnovu nje se određuje da li je promenljiva višedimenziona, da li je dimenzionisana i da li joj treba dodeliti standardne dimenzije.

SD (13) – VALTYP, 1 S
Ova promenljiva pokazuje da li je podatak koji se obrađuje string (vrednost SFF (255)) ili je broj (vrednost 0). Vrednost VALTYP se postavlja pri svakoj promeni vrednosti ili pri kreiranju promenljive.

SE (14) – INTFLG, 1 S
Tip numeričkog podatka: S80 (128) — ceo broj, 0 — broj sa pokretnim zarezom.

SF (15) – GARBFL, 1 S
Ova promenljiva obaveštava LIST rutinu da je naišao string pod znacima navoda i da ga treba ispisati bez konverzije tokena u string.
Ona takođe sadrži informaciju o tome da li se može nastaviti preuređivanje stringova (engl. garbage collection) u cilju oslobađanja nepotrebno zauzete memorije.

$10 (16) – SUBFLG, 1 S
Ovu promljivu koristi PTRGET rutina koja pronalazi ili kreira promenljive i to onda kada ispituje valjanost imena promenljive. Ukoliko je otvorena zagrada, SUBFLG se postavlja tako da pokazuje da li je promenljiva višedimenziona ili je to funkcija definisana od strane korisnika.

$11 (17) – INPFLG, 1 S
Način unošenja podataka: $98 (152) — READ, $40 (64) — GET, $0 (0) — INPUT.

S12 (18) – TANSGN, 1 S
Znak rezultata funkcija TAN i SIN. Druga funkcija joj je rezultat poređenja dve promenljive: 1 – ako je veće, 2 – ako je jednako i 4 – ako je manje. Kod kombinovanih operacija (>=,<=,<>) rezultat je kombinacija prethodnih brojeva.

S13 (19) – CHANNL, 1
Broj trenutnog ulaznog/izlaznog kanala. Ukoliko je ulazni uređaj 0 (tastatura), a izlazni 3 (ekran), ovaj broj je nula.

S14 (20) – LINNUM, 2
Broj linije koje koriste GOTO, LIST, ON ili GOSUB smešta se u ovu lokaciju. PEEK, POKE, WAIT i SYS koriste ovu rutinu kao pokazivač adrese na koju se naredba odnosi. Ova lokacija se nekad koristi i kao akumulator za cele dvobajtne brojeve.

S16 (22) – TEMPPT, 1 P
Pokazivač steka koji sadrži deskriptore privremenih stringova. Pošto ovaj stek sadrži svega tri deskriptora od po tri bajta, a počinje od S19 (25), broj S22 (34) znači da je stek pun pa ukoliko se pokuša dodavanje novog deskriptora, generiše se greška FORMULA TOO COMPLEX.

S17 (23) – LASTPT, 2 P
Pokazivač poslednjeg iskorišćenog segmenta u deskriptorskom steku. Njegova vrednost je za 3 manja od deskriptor stek pointera (koji pokazuje na prvi slobodan segment od 3 bajta).

S19 (25) – TEMPST, 9 R
Stek za deskriptore privremenih stringova. Privremeni string je onaj koji nije dodeljen string promenljivoj. Na primer „zdravo" u izkazu PRINT „zdravo". Svaki deskriptor sadrži 3 bajta. Prvi bajt je dužina stringa, a druga dva daju ofset adresu početka, odnosno, kraja stringa. U ovom steku ima prostora za tri deskriptora.

S22 (34) – INDEX, 4 P
Višenamenski prostor, najčešće dva pokazivača za premeštanje stringova i slično.

S26 (38) – RESHO, S R
Ovu oblast koriste rutine za množenje i deljenje brojeva sa pokretnim zarezom. Takođe se koristi i za rutine za računanje veličine oblasti memorije za smeštanje matrica.

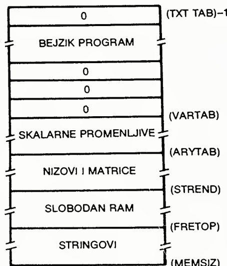
S2B (43) – TXTTAB, 2 P
Pokazivač početka bejzik teksta. Standardno pokazuje na S801 (2049).

S2D (45) – VARTAB, 2 P
Pokazivač početka bejzik promenljivih. Svakoj promenljivoj je dodeljeno sedam bajtova. Prva dva bajta sadrže ASCII vrednost prva dva karaktera imena promenljive. Ako ovo ime ima samo jedno slovo, drugi bajt je nula.
Sedmi bit jednog ili oba ova bajta može biti postavljen na jedinicu (ekvivalentno dodavanju S80 (128). U zavisnosti od toga određuje se tip promenljive.

1.bajt	2.bajt	promenljiva
+0	+0	vrednost sa pokretnim zarezom
+ S80	+0	string
+0	+ S80	funkcija (FN)
+ S80	+ S80	celobrojna vrednost

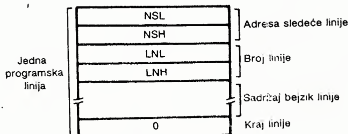
Upotreba poslednjih pet bajtova zavisi od tipa promenljive. Promenljive sa pokretnim zarezom koriste svih pet bajtova, dok celobrojne koriste samo treći i četvrti bajt, a preostala tri su neiskorišćena.
String promenljive koriste treći bajt za dužinu, a četvrti i peti kao pokazivače adrese odakle počinje string tekst, ostavljajući dva poslednja bajta neiskorišćena. String se može nalaziti u samom bejzik programu ili u oblasti na koju pokazuje FRETOP – S33 (51).

Definisana funkcija koristi treći i četvrti bajt kao pokazivač adrese bejzik programskog teksta gde započinje definicija funkcije. Peti i šesti bajt čine pokazivač nezavisno promenljive (npr. x u FN A(x)) dok je poslednji bajt neiskorišćen. Promenljive se smeštaju onim redom kojim se kreiraju, a pretraživanje promenljivih započinje od prve i ide redom do poslednje.
VARTAB se resetuje na jedan bajt više od adrese kraja bejzik programskog teksta uvek posle izvršenja naredbi CLR, NEW, RUN ili LOAD.
Ubacivanje novih programskih linija ili izmena starih, može izmeniti vrednosti promenljivih.



Na slici je prikazan deo memorijskog prostora u kome su smešteni bejzik program i bejzik promenljive. Prikazane su i odgovarajuće sistemske promenljive koje svojim vrednostima određuju raspodelu memorijskog prostora.

Na slici je prikazana struktura memorijskog prostora u kome je smeštena jedna bejzik programska linija.



$2F (47) — ARYTAB, 2 P
Pokazivač kraja oblasti za skalarne bejzik promenljive i početka oblasti za višedimenzione bejzik promenljive. Format zapisivanja je sledeći:

Prva dva bajta su rezervisana za ime. Format i značenje postavljenog sedmog bita je isti kao i za skalarne bejzik promenljive, s tim što ne postoji ekvivalent za definisanu funkciju.
Sledeća dva bajta čine ofset početka sledeće višedimenzione promenljive.
Zatim sledi jedan bajt koji sadrži broj dimenzija promenljive (npr. 2 za A(x,y)).
Iza ovoga se nalaze parovi bajtova od kojih svaki sadrži veličinu svake dimenzije uvećanu za 1 (jer postoji i indeks 0).
Na kraju dolaze vrednosti samih promenljivih. Format je isti kao i za skalarne promenljive, sem što one zauzimaju onoliko prostora koliko im je potrebno: promenljive sa pokretnim zarezom — pet bajtova, celobrojne — dva bajta, a deskriptori stringova — tri bajta. Naravno, sami stringovi se nalaze u nekom drugom delu memorije.

S31 (49) — STREND, 2 P
Pokazivač poslednje adrese uvećane za jedan koju zauzimaju višedimenzione promenljive i početak slobodnog RAM-a. Pošto se stringovi upisuju počev od najviše memorijske lokacije pa naniže, STREND označava poslednju moguću adresu za string. Ukoliko nema više prostora za stringove, obavlja se preuređivanje (engl. garbage collection) pa se oslobađa nepotrebno zauzeti deo memorije. Ovo se obavlja i izvršavanjem funkcije FRE.

S33 (S1) — FRETOP, 2 P
Pokazivač najniže adrese koju zauzimaju stringovi, što je ujedno i vrh slobodnog RAM-a.

S3S (S3) — FRESPC, 2 P
Pokazivač poslednjeg stringa dodatog u memoriji. Ovaj string se trenutno obrađuje.

S37 (SS) — MEMSIZ, 2 P
Pokazivač najviše memorijske adrese koju koristi bejzik. On se postavlja za vreme izvršavanja Kernal RESET rutine tj. posle ispitivanja RAM-a na vrednost S9FFF (40959). Ova vrednost se menja u dva slučaja:
— Kada se otvori RS232 kanal ona se pomera naniže za S12 bajtova, kreirajući dva bafera po 2S6 bajtova
— Kada je potrebno obezbediti sigurnu oblast u memoriji za ekransku memoriju, mašinski program ili slično.
Bejzik program se smešta u oblast memorije ograničene sadržajem VARTAB i TXTTAB. On uvek počinje nulom, a završava se sa dve nule.
Svaka linija započinje adresom sledeće linije (dva bajta), a zatim sledi broj linije. Iza njega se nalazi programski tekst sve dok se ne naiđe na nulu koja obeležava kraj linije. Naredbom NEW se dve nule koje označavaju kraj, upisuju odmah iza nule koja označava početak programa.

S39 (57) — CURLIN, 2
Ova lokacija sadrži broj linije koja se trenutno izvršava. Vrednost SFF (2SS) na lokaciji S3A (58) označava da se radi o direktnom načinu rada. U programskom načinu rada, ovde se upisuje broj linije pre izvršenja. Ovu lokaciju koriste STOP i END naredba kao i STOP taster za poruku o liniji u kojoj je prekinut program.

S3B (S9) — OLDLIN, 2
Kada se prekine izvršenje programa, broj poslednje izvršene linije se upisuje ovde. Ovaj broj se kopira u CURLIN naredbom CONT.

S3D (61) — OLDTXT, 2 P
Ovaj pokazivač sadrži adresu (ne broj linije) teksta bejzik izkaza koji se trenutno izvršava. Vrednost pokazivača TXTPTR — S7A (122) se ovde upisuje svaki put kada se počinje sa izvršavanjem nove bejzik linije. Naredbe END, STOP i pritisnut taster STOP upisuju u ovu lokaciju vrednost TXTPTR, dok se vrednost vraća u TXTPTR naredbom CONT.

S3F (63) — DATLIN, 2
Ova lokacija sadrži broj linije DATA iskaza koja se trenutno čita naredbom READ.

S41 (6S) — DATPTR, 2 P
Pokazivač adrese (ne broja linije) unutar bejzik teksta gde se podaci DATA čitaju naredbom READ
Naredba RESTORE vraća ovaj pokazivač na početnu adresu.

S43 (67) — INPPTR, 2 P
Pokazivač adrese izvora podataka za GET, INPUT (ulazni bafer — S200 (S12)) ili READ (DATA linija).

S45 (69) – VARNAM, 2
Pokazivač adrese gde se nalazi deskriptor ili vrednost promenljive koja se trenutno obrađuje. On pokazuje dva bajta iza imena promenljive.

S49 (73) – FORPNT, 2 P
Adresa indeksne promenljive koja se koristi u FOR petlji prvo se stavlja u ovaj pokazivač, a zatim se stavlja na stek. Ova lokacija se time oslobađa kao radna lokacija za INPUT, GET, READ, LIST, WAIT, CLOSE, SAVE, RETURN i GOSUB.

S4B (75) – OPPTR, 2 P
Adresa trenutno korišćenog operatora u operatorskoj tabeli OPTAB.

S4D (77) – OPMASK, 1
Rutina za računanje izraza na ovoj lokaciji kreira masku koja joj omogućuje da zna rezultat poređenja: manje (1), jednako (2), ili veće.

S4E (78) – DEFPNT, 2 P
Pokazivač deskriptora definisane funkcije (FN) koja se trenutno kreira.

S50 (80) – DSCPNT, 3 P
Prva dva bajta ovog pokazivača sadrže adresu trenutnog string deskriptora, dok treći bajt sadrži dužinu stringa.

S53 (83) – FOURG, 1
Konstanta koja se koristi pri preuređivanju stringova.

S54 (84) – JMPER, 3
Skok na rutinu odgovarajuće funkcije. Prvi bat je S4C (JMP naredba). Zatim sledi adresa funkcije dobijena iz tabele FUNDSP – SA052 (41042).

S57 (87) – WARE1, 10
Radni prostor bejzika.

S61 (97) – FAC1, 6 R
Akumulator za rad sa pokretnim zarezom 1 (engl. floating point accumulator 1).

S61 (97) – FACEXP, 1 R
Eksponent FAC 1. Eksponent se predstavlja kao 2^E pri čemu se E nalazi u ovoj lokaciji i to sa sledećim značenjem:

S80 (128) – označava E = 0
S81 (129) – označava E = 1
S82 (130) – označava E = 2 itd. sve do SFF (255)

Brojevi manji od 128 su negativni.
S7F (127) – označava E = –1
S7F (126) – označava E = –2 itd.

S62 (98) – FACHO, 4 R
Prvi bit sadrži znak broja, dok preostalih 31 bita čine normalizovanu mantisu broja (opseg od 0,5 do 1).

S66 (102) – FACSGN, 1 R
Znak broja u FAC1. 0 označava pozitivan broj, a SFF (255) označava negativan broj.

S67 (103) – SGNFLG, 1 B
Broj posebnih operacija koje je potrebno obaviti pri izračunavanju matematičke formule.

S68 (104) – BITS, 1 R
Indikator prekoračenja maksimalne vrednosti broja kod FAC1 (engl. overflow digit).

S69 (105) – FAC2, 6 R
Akumulator za rad sa pokretnim zarezom 2

S69 (105) – ARGEXP, 1 R
FAC 2 eksponent
S6A (106) – ARGHO, 4 R
FAC2 normalizovana mantisa
S6E (110) – ARGSGN, 1 R
FAC2 znak

$6F (111) – ARISGN, 1 R
Rezultat poređenja znakova FAC1 i FAC2. 0 označava da su znakovi jednaki, a $FF (255) označava da su znakovi različiti.

$70 (112) – FACOV, 1 R
Ukoliko mantisa broja ima više značajnih cifara nego što može da se smesti u četiri bajta, najmanje značajna cifra se smešta ovde. To se zatim koristi za povećanje tačnosti međurezultata i zaokruživanje krajnjeg rezultata.

$71 (113) – FBUFPT, 2 P
Pokazivač privremene tabele koja se koristi za računanje formula.

$73 (115) – CHRGET, 24 potprogram
Ovo je mašinski potprogram koji se u vreme inicijalizacije kopira sa adrese MOVCHG – $E3A2 (58274) u RAM. To je ključna rutina bejzik interpretera pomoću koje on čita programski tekst. Pokazivač adrese bajta koji se trenutno čita je, u stvari, operand LDA naredbe. Kada se u ovaj potprogram uđe preko adrese CHRGET, menja se vrednost operanda TXTPTR, omogućujući čitanje sledećeg karaktera. Ulaz preko adrese CHRGOT ne menja vrednost TXTPTR, pa se čita isti karakter. Dalje, rutina preskače prazna mesta i postavlja odgovarajuće bite u P registru mikroprocesora. Ako je C indikator (engl. carry flag) na logičkoj nuli, radi se o ASCII broju od 0 do 9, a ako je na jedinici, to je bilo koji drugi ASCII karakter. Ako je Z=1, pročitani karakter je znak kraja iskaza (0) ili dvotačka. U drugim slučajevima je Z=0.

```
$73 (115) CHRGET   INC TXTPTR; uvećaj sadržaj adrese TXTPTR
$75 (117)          BNE CHRGOT; za jedan (niži bajt)
$77 (119)          INC TXTPTR+1; uvećaj sadržaj adrese TXTPTR+1 za jedan (viši bajt)
$79 (121) CHRGOT   LDA; učitaj u akumulator broj sa adrese na koju pokazuje TXTPTR
$7A (122) TXTPTR   $0207; tj. neki iz tekst bafera $200–$250
$7C (124) POINTB   CMP $3A; postavi C indikator ako nije broj
$7E (126)          BCS EXIT;
$80 (128)          CMP $20; ako je prazno mesto, idi na
$82 (130)          BEQ CHRGET; sledeći karakter
$84 (132)          SEC; postavi C indikator ako nije nula
$85 (133)          SBC $30; ASCII ekvivalent brojeva od 0
$87 (135)          SEC; do 9 je $30–$39 (48–57)
$88 (136)          SBC $D0;
$8A (138) EXIT     RTS;
```

$8B (139) – RNDX, 5 R
Ovde se nalazi vrednost za računanje sledećeg slučajnog broja sa pokretnim zarezom. Po završetku računanja tu se smešta i sam slučajni broj.

$90 (144) – STATUS, 1 S
Kernal ulazno/izlazna status promenljiva. Za detalje pogledati READST rutinu.

$91 (145) – STKEY, 1 S
Promenljiva koja pokazuje da li je pri poslednjem ispitivanju bio pritisnut STOP taster (videti Kernal rutinu STOP). Ako je bio pritisnut, vrednost je $7F (127).

$92 (146) – SVXT, 1 B
Vremenska kontrola učitavanja sa kasetofona.

$93 (147) – VERCK, 1 S
Isto kao i na adresi $A (10).

$94 (148) – C3PO, 1 S
Pokazuje da postoji karakter u baferu koji čeka na slanje serijskom vezom.

$95 (149) – BSOUR, 1 R
Ovde se smešta karakter koji se šalje preko serijske veze. Broj $FF (255) znači da nema karaktera za slanje.

$96 (150) – SYNO, 1 R
Broj za sinhronizaciju kasetnog bloka

$97 (151) – XSAV, 1 R
Lokacija gde se čuva sadržaj X registra.

S98 (1S2) – LDTND, 1 P
Broj trenutno otvorenih datoteka (maksimalno 10). OPEN povećava sadržaj za jedan, a CLOSE smanjuje za jedan. Kernal rutina CLALL upisuje u LDTND nulu. Sadržaj ove lokacije se koristi kao indeks za kraj tabela gde se nalaze brojevi datoteka, primarne i sekundarne adrese – LAT – S2S9 (601), FAT – S263 (611) i SAT – S26D (621).

S99 (1S3) – DFLTN, 1
Broj trenutnog ulaznog uređaja. Po inicijalizaciji to je 0 (tastatura). Može se menjati Kernal rutinom CHKIN.

S9A (1S4) – DFLTO, 1
Broj trenutnog izlaznog uređaja. Po inicijalizaciji to je 3 (ekran), ali se može trajno promeniti naredbom CMD. Takođe se može menjati Kernal rutinom CHKOUT.

S9B (1SS) – PRTY, 1 R
Lokacija za ispitivanje parnosti primljenog bajta sa kasetofona.

S9C (1S6) – DPS=, 1 S
Promenljiva koja ukazuje da li je delimično ili potpuno primljen bajt sa kasetofona.

S9D (1S7) – MSGFLG, 1 S
Kontrola ispisivanja poruka Kernala. Za detalje pogledati Kernal rutinu SETMSG.

S9E (1S8) – PTR1, 1 S
Indikator greške u parnosti pri čitanju prvog zapisa sa kasetofona.

S9F (1S9) – PTR2, 1 R
Lokacija koja se koristi za korekciju pogrešno učitanog bajta u prvom čitanju zapisa sa kasetofona.

SA0 (160) – TIME, 3 R
Softverski časovnik realnog vremena. Videti Kernal rutine SETTIM, RDTIM i UDTIM.

SA3 (163) – XXXX, 2
Lokacije za privremeno smeštanje podataka.

SAS (16S) – CNTDN, 1 B
Brojač sinhronizacionih karaktera koji se šalju neposredno pre podataka na kasetofon.

SA6 (166) – BUFPNT, 1 B
Brojač broja bajtova koji se čitaju ili upisuju u kasetni bafer. Podaci se fizički šalju na kasetofon tek kada ovaj brojač izbroji 192 bajta, tj. kada je bafer pun.

SA7 (167) – INBIT, 1 R
U ovoj lokaciji se privremeno nalaze biti sa RS 232 ulazne linije, ali se ona koristi i za razne operacije pri radu sa kasetofonom.

SA8 (168) – BITCI, 1 B
Brojač primljenih bita, u okviru jedne reči, sa RS 232 veze. Takođe se koristi kao pokazivač greške prilikom operacija čitanja sa kasetofona.

SA9 (169) – RINONE, 1 S
Pokazivač nailaska start bita sa RS 232 veze. Broj S90 (144) označava da start bit nije primljen, dok S0 (0) pokazuje da je primljen.

SAA (170) – RIDATA, 1 S
Bafer primljenog karaktera preko RS 232 veze. Karakter se zatim stavlja u RS 232 bafer na koji pokazuje RIBUF – SF7 (247). Lokacija se takođe koristi za određivanje razlike podataka i sinhronizacionih karaktera pri radu sa kasetofonom.

SAB (171) – RIPRTY, 1 S
Ova lokacija se koristi pri proveri parnosti podataka primljenih preko RS 232 veze. Služi i kao indikator kompletnosti zaglavlja pri radu sa kasetofonom.

SAC (172) – SAL, 2 P
Pokazivač početka RAM-a odakle se vrši SAVE ili LOAD rutine. Takođe se koristi i pri izvršavanju SCROLL rutine.

SAE (174) – EAL, 2 P
Ovu lokaciju koristi Kernal rutina SAVE da bi pokazala krajnju adresu za SAVE, LOAD ili VERIFY.

$B0 (176) – CMPO, 2
Pomoćna lokacija za određivanje vrednosti vremenske konstante u SVHT – $92 (146).

$B2 (178) – TAPE1, 2 P
Pokazivač početka kasetnog bafera. Mora da sadrži broj veći od $200 (512), inače će se pri pokušaju rada sa kasetofonom javiti poruka ILLEGAL DEVICE NUMBER. Posle inicijalizacije pokazivač sadrži vrednost $33C (828).

$B4 (180) – BITTS, 1 B
Brojač poslatih bita preko RS 232 veze

$B5 (181) – NXTBIT, 1 S
Ova lokacija sadrži bit koji će kao sledeći biti poslat preko RS 232, a koristi se i kao indikator da je sa kasetofona primljen EOT marker.

$B6 (182) – RODATA, 1 R
Bafer za bajt koji se šalje preko RS 232 veze.

$B7 (183) – FNLEN, 1 R
Broj karaktera u nazivu datoteke. Pri radu sa diskom broj karaktera može biti od 1 do 16, kod kasetofona od 0 do 187 dok kod RS 232 može da bude od 0 do 4.

$B8 (184) – LA, 1 R
Logički broj datoteke koja se trenutno obrađuje. Ovaj broj može biti od 1 do 255. Maksimalni broj otvorenih datoteka može biti deset od čega maksimalno 5 za rad sa diskom.

$B9 (185) – SA, 1 R
Sekundarna adresa vezana za LA. Može biti od 0 do 31 za rad sa uređajima preko serijske veze i od 0 do 127 za ostale.

$BA (186) – FA, 1 R
Primarna adresa (broj uređaja) vezana za LA.

$BB (187) – FNADR, 2 P
Pokazivač (adresa) naziva datoteke.

$BD (189) – ROPRTY, 1 R
Lokacija koja se koristi pri slanju bita parnosti preko RS 232 veze.

$BE (190) – FABLK, 1 B
Brojač preostalih blokova koji treba da se pošalju ili prime sa kasetofona.

$BF (191) – MYCH, 1 R
Bafer primljenog bajta pri čitanju sa kasetofona.

$C0 (192) – CAS1, 1 S
Kontrola rada motora kasetofona. U okviru rutine za obradu IRQ, tastira se sadržaj ove lokacije kao i to da li je pritisnut taster na kasetofonu. Ako je taster pritisnut i pri tome CAS1 sadrži 0, motor se uključuje.

$C1 (193) – STAL, 2 P
Pokazivač lokacije u RAM-u odakle je obavljena operacija LOAD ili SAVE. On će pokazivati na kasetni bafer, pri radu sa kasetofonom, dok će se za ostale operacije koristiti oblast RAM-a na koju pokazuje MEMUSS.

$C3 (195) – MEMUSS, 2 P
Videti lokaciju STAL.

$C5 (197) – LSTX, 1 R
Redni broj (koordinata) tastera koji je bio pritisnut za vreme izvršavanja poslednje IRQ rutine. Broj $40 (64) znači da nijedan taster nije bio pritisnut.

$C6 (198) – NDX, 1 R
Broj karaktera koji se nalaze u baferu tastature (engl. keyboard queue). Maksimalni broj je određen sadržajem XMAX – $289 (649), a po inicijalizaciji to je 10.

$C7 (199) – RVS, 1 R
Kada se pod znacima navoda pritisnu tasteri CTRL i RVS-ON (CHR$ (18)), ovaj indikator se postavlja

na $12 (18) Rutine za prikazivanje na ekranu na osnovu ovoga dodaju $80 (128) na ekranski kôd svakog karaktera, tako da se on prikazuje u inverznom obliku

$C8 (200) — INDX, 1 P
Pokazivač poslednjeg karaktera (različitog od praznog mesta) u logičkoj liniji koji se unosi preko INPUT

$C9 (201) — LXSP, 2 R
Koordinata (broj linije, broj kolone) kursora na početku INPUT

$CB (203) — SFDX, 1 R
Redni broj (koordinata) poslednjeg pritisnutog tastera Ovaj broj se koristi kao indeks za dekodovanje iz jedne od ASCII dekoderskih tabela

$CC (204) — BLNSW, 1 S
Kontrola kursora Ukoliko je vrednost 0, kursor trepće

$CD (205) — BLNCT, 1 B
Brojač trajanja jednog treptaja kursora Prvo se postavlja na vrednost 20 pa se umanjuje svakih 1/60 sekunde. Kada se dostigne 0, menja se stanje kursora, ponovo se postavlja broj 20 itd. Na osnovu ovoga kursor trepće tri puta u sekundi.

$CE (206) — GDBLN, 1 R
Ekranski kôd karaktera koji je trenutno „ispod" kursora

$CF (207) — BLNON, 1 S
Stanje kursora u jednom ciklusu treptanja: 0 — kursor se ne vidi, 1 — kursor se vidi.

$D0 (208) — CRSW, 1 S
Ovu lokaciju koristi Kernal CHRIN rutina. Ona na osnovu njenog sadržaja određuje da li se učitavanje vrši sa ekrana (3) ili sa tastature (0).

$D1 (209) — PNT, 2 P
Adresa u ekranskoj memoriji linije u kojoj se nalazi kursor.

$D3 (211) — PNTR, 1 P
Broj kolone u ovoj liniji u kojoj je kursor. Može biti i veći od 40 ako su dve fizičke linije povezane u jednu logičku.

$D4 (212) — QTSW, 1 S
Indikator ispisa neparnog broja znakova navoda. Ukoliko je vrednost različita od nule, editor smatra da je ispisan neparan broj znakova navoda (engl. quote mode) i tada kontrolni karakteri (osim CHR$ (13), CHR$ (141) i CHR$ (20)) gube svoju funkciju.

$D5 (213) — LNMX, 1 R
Maksimalna dužina fizičke linije.

$D6 (214) — TBLX, 1 P
Ova lokacija sadrži broj fizičke linije (0 – 24) u kojoj je kursor.

$D7 (215) — XXXX, 1
ASCII vrednost poslednjeg karaktera ispisanog na ekranu.

$D8 (216) — INST, 1 S
Kada se pritisne taster INST, ekranski editor pomera desni deo linije za jedno mesto, dodaje novu fizičku liniju trenutnoj logičkoj (ukoliko je potrebno), povećava LNMX $D5 (213) za jedan i ažurira link tabelu LDTB1. Lokacija INST sadrži broj praznih mesta koja su stvorena pritiskanjem tastera INST.

$D9 (217) — LDTB1, 26 T
Ekranska link tabela. Svakom bajtu tabele odgovara po jedna fizička linija. Bit 0 do 3 svakog bajta određuju na kojoj se stranici ekranske memorije nalazi prvi bajt dotične linije. Ekranska memorija ima 1kB što je četiri stranice po 256 bajtova.

Svaki bit svakog bajta određuje povezanost fizičkih linija. Ako je taj bit jedinica, ta linija je prva ili jedina fizička linija. Ukoliko je nula, ta linija je druga polovina logičke linije.

F3 (243) — USER, 2 P
Adresa prvog bajta u kolor RAM-u. Svaka lokacija kolor memorije ima odgovarajuću lokaciju u ekranskoj memoriji.

$F5 (245) – KEYTAB, 2 P
Pokazivač ASCII dekodreske tabele koja se koristi zavisno od pritisnutog tastera SHIFT, CTRL i C=
Za detalje treba pogledati opis Kernal SCNKEY rutine.

$F7 (247) – RIBUF, 2 P
Kada se otvori datoteka na uređaju 2 (RS 232 veza), na kraju memorije predviđene za bejzik, stvaraju se dva bafera po 256 bajtova. Ova lokacija sadrži adresu bafera koji je ulazni.

$F9 (249) – ROBUF, 2 P
Adresa izlaznog bafera od 256 bajtova za rad preko RS 232 veze.

$FB (251) – FREKZP, 4
Slobodne lokacije na nultoj strani koje bejzik sigurno ne koristi.

$FF (255) – BASZPT, 1
Privremena lokacija pri konvertovanju brojeva sa pokretnim zarezom u ASCII karaktere.

$100 – $1FF (256 – 511)
Mikroprocesorski stek. Osim standardnih operacija sa stekom, ovaj deo memorije se koristi i za sledeće operacije.

$100 – $10A (256 – 266)
Radna oblast za konverziju brojeva u ASCII cifre.

$100 – $13E (256 – 318) – BAD
Svaki blok se na kasetofonu snima dva puta radi kasnije korekcije eventualne greške. Ovih 62 bajta sadrže indekse onih bajtova koji nisu učitani ispravno za vreme prvog čitanja, tako da se može izvršiti popravka za vreme drugog čitanja.

$13F – $1FF (319 – 511)
Ova oblast se koristi isključivo kao mikroprocesorski stek. Za detalje o njegovom korišćenju pogledati poglavlje 7. Bejzik vrlo iscrpno koristi stek. Pre njegove upotrebe, on proverava da li je na njemu najmanje 62 slobodna bajta. Ukoliko nije, javlja se poruka OUT OF MEMORY.
Svaka FOR naredba zahteva 18 bajtova na steku.
Prvo dolazi konstanta $81 (129), a zatim dvobajtna adresa indeksne promenljive (npr. X u iskazu FOR X = 1 TO 10). Za njima sledi petobajtni broj (sa pokretnim zarezom) – vrednost za STEP kao i petobajtni broj za krajnju vrednost indeksa (TO). Na kraju dolazi dvobajtni broj linije na koju se program vraća posle NEXT, kao i dvobajtna adresa sledećeg karaktera u toj liniji posle FOR iskaza.
Svaka GOSUB naredba zahteva 5 bajtova na steku.
Prvi je konstanta $8D (141), a zatim sledi dvobajtni broj linije na koju se program vraća posle naredbe RETURN. Poslednja dva bajta čine pokazivač programskog teksta po povratku iz potprograma.
Svaka DEF naredba zahteva takođe 5 bajtova na steku.
Njihova funkcija je identična kao i kod GOSUB osim što prvi bajt nema nikakvog značaja.

$200 – $258 (512 – 600) – BUF, 89 R
Kada se radi u direktnom načinu rada, karakteri jedne linije koji se unose sa tastature smeštaju se u ovu oblast memorije. Bejzik interpreter čita ove karaktere i konvertuje odgovarajuće grupe u tokene. Zatim se cela linija smešta u memoriju ili izvršava zavisno od toga da li počinje brojem ili ne.
Bafer je veličine 89 bajtova, a ekranski editor dozvoljava maksimalnu dužinu linije od 80 karaktera plus jedan bajt za nulu koja označava kraj linije. Prema tome, poslednjih 8 bajtova je slobodno za korisnika.

$259 (601) – LAT, 10 T
Tabela trenutno otvorenih logičkih datoteka.

$263 (611) – FAT, 10 T
Tabela primarnih adresa (brojeva uređaja) vezanih za LAT.

$26D (621) – SAT, 10 T
Tabela sekundarnih adresa vezanih za LAT.

$277 (631) – KEYD, 10 R
Za vreme izvršavanja prekidne rutine (IRQ) u ovaj bafer se smešta ASCII kôd tastera koji je bio pritisnut. Bafer je organizovan kao red (engl. queue) tj. on radi na principu FIFO registra (engl. first in first out)

tj. onaj bajt koji je prvi ušao u bafer prvi će i izaći. Bejzik uzima jedan po jedan karakter koji smešta u BUF i ispisuje na ekranu. U baferu može biti najviše 10 karaktera (ovo je određeno sadržajem XMAX – $289 (649)). To omogućuje brže unošenje karaktera preko tastature nego što se oni ispisuju na ekranu.

$281 (641) – MEMSTR, 2 P
U ovu lokaciju Kernal rutina RAMTAS upisuje adresu najniže RAM lokacije raspoložive za bejzik programe. Posle inicijalizacije to je $800 (2048).

$283 (643) – MEMSIZ, 2 P
Posle nedestruktivnog testiranja memorije, RAMTAS rutina upisuje ovde adresu najviše lokacije raspoložive za bejzik programe.

$285 (645) – TIMOUT, 1 S
Koristi se uz dodatnu IEEE 488 karticu. Za detalje pogledati Kernal rutinu SETTMO.

$286 (646) – COLOR, 1 R
Broj boje kojom se vrši ispisivanje karaktera na ekranu.

$287 (647) – GDCOL, 1 P
Boja karaktera preko koga je trenutno postavljen kursor.

$288 (647) – HIBASE, 1 P
Gornji bajt apsolutne adrese gde počinje ekranska memorija koja zauzima 1024 bajta. Posle inicijalizacije, ekranska memorija počinje od $400 (1024) ali se ova vrednost može menjati. Pri tome mora da se promeni i vrednost na lokaciji $DD00 (5676) i $D018 (53272). To su adrese registara video kontrolera.

$289 (649) – XMAX, 1 S
Broj koji označava maksimalni broj karaktera koji se mogu istovremeno naći u KEYD.

$28A (650) – RPTFLG, 1 S
Kontrola automatskog ponavljanja pritisnutog tastera. Standardna vrednost je 0 što znači da se ponavljaju samo tasteri za kontrolu kursora, SPACE i INST/DEL taster. Vrednost $80 (128) znači da se svi tasteri mogu ponavljati dok vrednost $40 (64) znači nijedan.

$28B (651) – KOUNT, 1 B
Brojač vremena koje treba da prođe da bi taster počeo da se ponavlja. Pri inicijalizaciji DELAY sadrži broj 16. Sve dok se taster drži pritisnut, ova vrednost se umanjuje za 1 svakih 1/60 sekunde. Kada dostigne 0, KOUNT počinje sa odbrojavanjem od svoje početne vrednosti tj. od 6. Kada KOUNT dostigne 0, ukoliko RPTFLG to dozvoljava, počinje ponavljanje, KOUNT se postavlja na 4, odbrojava do nule itd. Prema tome, pre ponavljanja se čeka 22/60 sekunde, a zatim sledi ponavljanje 15 puta u sekundi.

$28D (653) – SHFLAG, 1 S
Ova promenljiva sadrži brojeve koji imaju sledeće značenje:

01 – pritisnut je taster SHIFT
02 – pritisnut je taster C=
04 – pritisnut je taster CTRL

Ukoliko je pritisnuto više od jednog tastera, rezultat je zbir prethodnih vrednosti. Na primer, C= i SHIFT daju broj 03

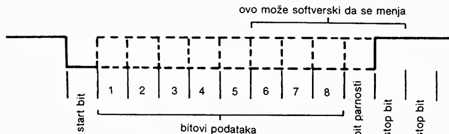
$28E (654) – LSTSHF, 1 S
Poslednja vrednost promenljive SHFLAG. Ovo omogućuje da se pritiskom na pojedinačne tastere SHIFT i C= ne menja skup karaktera.

$28F (655) – KEYLOG, 2 V
Vektor rutine operativnog sistema koja postavlja pokazivač KEYTAB na osnovu sadržaja SHFLAG. Pogledati Kernal rutinu SCNKEY.

$291 (657) – MODE, 1 S
Promenljiva koja dozvoljava ili onemogućuje promenu skupa karaktera istovremenim pritiskom na SHIFT i C=. Vrednost 0 znači dozvolu, a $80 (128) znači zabranu.

$292 (658) – AUTODN, 1 S
Vrednost 0 u ovoj lokaciji dozvoljava pomeranje sadržaja ekrana za jednu logičku liniju na gore (engl. scroll). Kada u baferu tastature ima karaktera za obradu, ovo se privremeno zabranjuje.

$293 (659) – M51CTR, 1 R
Pseudo 6551 kontrolni registar.
U Komodoru postoji softverska imitacija UART-a 6551 (engl. universal asynchronous receiver – transmitter) koji je namenjen serijskom prenošenju podataka preko RS 232 veze. Pojedinim registrima UART-a odgovaraju memorijske lokacije u Komodoru. Hardverski izvodi se nalaze na korisničkom priključku (pogledati poglavlje 12. Konstrukcije).
Sadržaj kontrolnog registra 6551 određuje brzinu prenosa (engl. baud rate), broj bita po jednom karakteru kao i broj STOP bita pri slanju poruke.



Sl. 8. 2. Način slanja bajta serijskom vezom

Bit 7 registra M51CTR određuje broj STOP bita:
0 (vrednost 0) – 1 STOP bit
1 (vrednost 128) – 0 STOP bita

Treba napomenuti da jedan STOP bit uvek ide i da se prethodno dodaje na njega.
Bit 6 i 5 određuju dužinu karaktera podataka:
00 (vrednost 0) – 8 bita podataka
01 (vrednost 32) – 7 bita podataka
10 (vrednost 64) – 6 bita podataka
11 (vrednost 96) – 5 bita podataka
Bit 4 se ne koristi.
Bit 3 do 0 određuju brzinu prenosa (engl. baude rate):

0000 (vrednost 0) – Brzinu definiše korisnik.
0001 (vrednost 1) – 50 Baud
0010 (vrednost 2) – 75 Baud
0011 (vrednost 3) – 110 Baud
0100 (vrednost 4) – 134.5 Baud
0101 (vrednost 5) – 150 Baud
0110 (vrednost 6) – 300 Baud
0111 (vrednost 7) – 600 Baud
1000 (vrednost 8) – 1200 Baud
1001 (vrednost 9) – 1800 Baud
1010 (vrednost 10) – 2400 Baud

Iako u originalnom 6551 UART-u postoje brzine 3600, 4800, 7200, 9600 i 19200 Bauda, ovde nisu implementirane jer ne mogu da se dobiju softveski.

$294 (660) – M51CDR, 1 R
Psuedo 6551 komandni registar. Njegov sadržaj određuje tip parnosti, tip veze i postojanje kontrolnih signala (engl. handshake).
Biti 7 do 5 – Tip parnosti.

XX0 (vrednosti 0, 64, 128 ili 192) — bez parnosti
001 (vrednost 32) — neparna parnost
011 (vrednost 96) — parna parnost
101 (vrednost 160) — prenos jedinice na mestu bita parnosti
111 (vrednost 224) — vrednost nule na mestu bita parnosti

Parnost se koristi za proveru ispravnosti prenete poruke. Pri slanju se sabiraju sve jedinice. Na osnovu toga da li je zbir paran ili neparan, generiše se bit parnosti tako da ukupan zbir (sa bitom parnosti) bude na primer paran (parna parnost). Pri prijemu se proverava da li je zbir jedinica paran broj. Ukoliko nije, verovatno je jedan bit pogrešno primljen.

Bit 4 — tip veze
0 (vrednost 0) — Full duplex. Podaci se šalju i primaju istovremeno.
1 (vrednost 16) — Half duplex. Podaci u jednom trenutku mogu da se šalju ili da se primaju.
Biti 3 do 1 — Ne koriste se.
Bit 0 — Postojanje kontrolnih signala.
0 (vrednost 0) — Nema kontrolnih signala. Prenos se obavlja preko tri linije: za slanje, za prijem i masa
1 (vrednost 1) — Koriste se svi kontrolni signali. (pogledati poglavlje 12. Konstrukcije).

S295 (661) — M51AJB, 2 R
Ova lokacija sadrži nestandardnu brzinu prenosa ukoliko je ona zadata pri otvaranju RS 232 kanala. Vrednost koja se ovde upisuje mora da bude 0.98525MHz/(Baud rate/2)—100 za PAL sistem.

S297 (663) RSSTAT, 1 S
Pseudo 6551 status registar. Može se pročitati direktno (pomoću PEEK) i koristeći bejzik promenijivu ST ili pomoću Kernal rutine READST. U poslednja dva slučaja sadržaj registra se briše pri čitanju.

S298 (664) — BITNUM, 1 B
Sadržaj lokacije određuje koliko nula treba dodati da bi se dobila dužina karaktera zadana u M51CTR. Ukupan broj bita treba da bude 8.

S299 (665) — BAUDOF, 2 R
Vrednosti koje se upisuju u registre A i B CIA #2 za dobijanje odgovarajućih brzina prenosa. One se računaju pomoću formule date u opisu lokacije M51AJB.

S29B (667) — RIDBE, 1 P
Pokazivač trenutno poslednjeg bajta u RS 232 prijemnom baferu.

S29C (668) — RIDBS, 1 P
Pokazivač trenutno prvog bajta u RS 232 prijemnom baferu.

S29D (669) — RODBS, 1 P
Pokazivač trenutno prvog bajta u RS 232 predajnom baferu.

S29E (670) — RODBE, 1 P
Pokazivač trenutno poslednjeg bajta u RS 232 predajnom baferu.

S29F (671) — IRQTMP, 2 P
Ovde se privremeno čuva IRQ vektor CINV — S314 (788) za vreme rada sa kasetofonom pri čemu se koristi posebna IRQ rutina.

S2A1 (673) — ENABL, 1 S
Ova lokacija sadrži vrednost NMI indikatora kontrolnog registra prekida iz CIA #2 — SDD0D (S6589). Pojedini biti imaju sledeće značenje:
Bit 4 : 1 (vrednost 16) — sistem čeka
Bit 1 : 1 (vrednost 2) — sistem prima podatke
Bit 0 : 1 (vrednost 1) — sistem šalje podatke

S2A2 — S2A5 (674 — 677), 4
Radni prostor pri komunikaciji sa kasetofonom. Privremeno čuvanje registara CIA #1 i slično.

$2A6 (678) – PALNTC, 1 S
Indikator sistema: 0 = NTSC, 1 = PAL.

$2A7 – 2FF (679 – 764) – XXXX, 89
Slobodne lokacije.

$300 (768) – IERROR, 2 V
Vektor rutine za obradu grešaka u bejziku. Po inicijalizaciji pokazuje na $E38B (58251).

$302 (770) – IMAIN, 2 V
Vektor koji pokazuje na početak bejzik interpreterske petlje. Po inicijalizaciji to je adresa $A483 (42115).

$304 (772) – ICRNCH, 2 V
Vektor bejzik rutine za tokenizaciju bejzik teksta. Po inicijalizaciji to je $A57C (42364).

$306 (774) – IQPLOP, 2 V
Vektor bejzik rutine za ispisivanje tokena u obliku niza ASCII karaktera. Po inicijalizaciji to je $A71A (42778).

$308 (776) – IGONE, 2 V
Vektor bejzik rutine koja izvršava sledeću naredbu. Po inicijalizaciji to je $A7E4 (42980).

$30A (778) – IEVAL, 2 V
Vektor bejzik rutine koja konvertuje aritmetički term u ekvivalent sa pokretnim zarezom. Po inicijalizaciji sadržaj IEVAL je $AE86 (44678).

$30C (780) – SAREG, 1 R
Pre izvršavanja bejzik naredbe SYS, akumulator (registar A) se puni sadržajem ove lokacije. Po povratku u Bejzik ovde se upisuje poslednji sadržaj akumulatora mašinskog potprograma. Prema tome SAREG se može koristiti za prenos parametara između bejzik programa i mašinskog potprograma.

$30D (781) – SXREG, 1 R
Isto kao SAREG samo se odnosi na registar X.

$30E (782) – SYREG, 1 R
Isto kao SAREG samo se odnosi na registar Y.

$30F (783) – SPREG, 1 R
Isto kao SAREG samo se odnosi na registar P.

$301 (784) – USRPOK, 1
Vrednost $4C (76) što je kôd za naredbu JMP (indikretno). Vektor skoka je USRADD.
$311 (785) – USRADD, 2 V
Vektor rutine USR funkcije. Pri inicijalizaciji on sadrži adresu bejzik rutine za obradu greške. Ukoliko se on programski ne promeni (tj. pozove se USR funkcija bez definisanja adrese odgovarajuće rutine) ispisuje se poruka ILLEGAL QUANTITY ERROR.

$313 (787) – XXXX
Slobodna lokacija.

$314 (788) – CINV, 2 V
Vektor standardne IRQ rutine. Po inicijalizaciji sadrži $EA31 (59953).

$316 (790) – CBINV, 2 V
Vektor rutine za obradu BRK prekida. Po inicijalizaciji sadrži $FE66 (65126).

$318 (792) – NMINV, 2 V
Vektor standardne NMI rutine. Po inicijalizaciji sadrži $FE47 (65095).

$31A (794) — IOPEN, 2 V
Vektor Kernal OPEN rutine. Standardno $F34A (62282).

$31C (796) — ICLOSE, 2 V
Vektor Kernal CLOSE rutine. Standardno $F291 (62097).

$31E (798) — ICHKIN, 2 V
Vektor Kernal CHKIN rutine. Standardno $F20E (61966).

$320 (800) — ICKOUT, 2 V
Vektor Kernal CHKOUT rutine. Standardno $F250 (62032).

$322 (802) — ICLRCH, 2 V
Vektor Kernal CLRCHN rutine. Standardno $F333 (62259).

$324 (804) — IBASIN, 2 V
Vektor Kernal CHRIN rutine. Standardno $F157 (61783).

$326 (806) — IBSOUT, 2 V
Vektor Kernal CHROUT rutine. Standardno $F1CA (61898).

$328 (808) — ISTOP, 2 V
Vektor Kernal STOP rutine. Standardno $F6ED (63213).

$32A (810) — IGETIN, 2 V
Vektor Kernal GETIN rutine. Standardno $F13E (61758).

$32C (812) — ICLALL, 2 V
Vektor Kernal CLALL rutine. Standardno $F32F (62255).

$32E (814) — USRCMD, 2 V
Vektor rutine za izvršavanje komande definisane od strane korisnika. Ovo je korišćeno kod PET računara za definisanje novih komandi monitor programa (kod PET-a je on u ROM-u). Ne može se koristiti za dodavanje novih naredbi. Po inicijalizaciji pokazuje ne rutinu za obradu BRK.

$330 (816) — ILOAD, 2 V
Vektor Kernal LOAD rutine. Standardno $F49E (62622).

$332 (818) — ISAVE, 2 V
Vektor Kernal SAVE rutine. Standarno $F5DD (62941).

$334 — $33B (820 — 827) — XXXX, 8
Slobodne lokacije.

$33C — $3FB (828 — 1019) — TBUFFER, 192 R
Bafer za rad sa kasetofonom. Tipovi blokova koji se čitaju sa kasete i koji se ovde smeštaju su: zaglavlje programa, zaglavlje podataka i sami podaci.
Prvi bajt bilo kog bloka (smešta se na $33C (828) identifikuje njegov tip. Blokovi zaglavlja slede odmah iza identifikacionog bajta i sadrže dva bajta početne RAM adrese, dva bajta krajnje RAM adrese i ime datoteke popunjeno blanko znacima tako da mu je ukupna dužina 187 bajtova. Blokovi podataka imaju 191 bajt iza identifikatora.
Vrednosti identifikatora mogu biti:

1 — Označava blok relokativne programske datoteke.
2 — Označava blok podataka.
3 — Označava blok nerelokativne programske datoteke.

Programske datoteke koriste TBUFFER samo za smeštanje zaglavlja dok se sam program puni direktno u memoriju od početka relokativne adrese I od nerelokativne adrese $801.

4 — Označava blok zaglavlja podataka. Za razliku od programskih datoteka, blokovi zaglavlja podataka i blokovi podataka se smeštaju u TBUFFER.

Podacima se pristupa bejzik komandama PRINT # za upis, INPUT # i GET # za čitanje.
5 – Označava da je to poslednji blok tj. lokacija kraja trake. Na osnovu toga Kernal prestaje sa učitavanjem iako se iza nalazi još podataka.

3FC – $3FF (1020 – 1023) XXXX.
Slobodne lokacije.

8. 3 BEJZIK INTERPRETER

Osnovna funkcija interpretera je da izvrši sukcesivne iskaze izvornog programskog jezika prevodeći ih direktno u operacije. Za svaki iskaz obavlja se istovetna sekvenca:
1. Uzimanje iskaza sa pozicije na koju pokazuje brojač iskaza
2. Uvećavanje sadržaja brojača
3. Analiza iskaza i određivanje operacije
4. Izvršavanje operacije

Odmah se može videti sličnost sa ciklusom na nivou mašinskog jezika gde se obavlja sledeća sekvenca:

1. Uzimanje naredbe sa lokacije na koju ukazuje programski brojač.
2. Uvećavanje sadržaja programskog brojača na vrednost adrese sledeće naredbe.
3. Dekodovanje naredbe.
4. Izvršavanje naredbe.

Prema tome, u mikroprocesuru postoji „hardverski" interpreter mikrokoda. I u jednom i u drugom slučaju postoji jedna petlja unutar koje se neprekidno obavljaju navedene četiri operacije, odnosno, mikro operacije. Takva petlja zove se interpreterska ili kontrolna petlja.

Kod bejzik interpretera svakom iskazu odgovara neki potprogram koji se bira unutar kontrolne petlje. Po izvršavanju potprograma, kontrola se vraća petlji i tako se nastavlja sve dok se ne izvrši prirodni prekid ili dođe do greške. Osim ovoga, vrši se selekcija operanada, a ako su oni zadati u simboličkom obliku, vrši se ažuriranje tabele simbola. Kontrolna petlja poziva, kao pomoć, i potprograme za leksičku i sintakstičku analizu programskog teksta. Leksička analiza omogućuje razlaganje izvornog programskog teksta na elemente od kojih je programski jezik sastavljen.

Sintaksička analiza određuje strukturne odnose među tim elementima.

Osnovna jedinica programskog teksta je karakter. Više grupisanih karaktera čine token, najmanju jedinicu programskog jezika koja ima neko značenje. Leksička analiza, prema tome, deli programski tekst na tokene i identifikuje njihov tip. Token može biti:

a. Službena reč. Obično se cela reč pri unošenju u memoriju koduje jednim bajtom.
b. Identifikator. U bejziku je to ime skalarne ili vektorske promenljive.
c. Konstanta
d. String
e. Operator: +, –, *, /, i slično.
f. Separator: :, ;, ⟨CR⟩, i slično.

Prazna mesta, komentare i slično interpreter ne uzima u obzir.

Identifikator je simbolička adresa gde se nalazi vrednost promenljive. Ukoliko je u pitanju string promenljiva, njen identifikator daje adresu gde se nalazi deskriptor stringa. Deskriptor sačinjavaju početna adresa stringa i njegova dužina.

Tokeni su grupisani u složenije celine:

a. Faktor (engl. factor). On uključuje promenljive, konstante bez predznaka, oznake funkcija, negacije faktora i izraze u zagradama.
Primer: ×, 12, sin (z), NOTP

b. Term (engl. term) On uključuje faktore i izraze oblika $T \otimes F$ gde je T term, F faktor, a jedan od operatora *, / i AND.
Primer: A * B, I/(I − 1), PANDQ

c. Prost izraz (engl. simple expression). Uključuje sve terme i izraze oblika $E \oplus T$, +T i −T gde je E prost izraz, T term, a $\oplus$ je jedan od operatora +, − i OR.
Primer: × + Y, B* B − 4* A* C

d. Izraz (engl. expression). To je struktura oblika E1ⓡE2 gde su E1 i E2 prosti izrazi, aⓡje jedan od racionalnih operatora <, <=, =, <>, >=, >.
Primer: × =1.5, B*B− 4 *A * C <= 0

e. Iskaz (engl. statement). Ovo je osnovna „rečenica" programskog jezika. Iskaz menja vrednost promenljivoj na osnovu izračunatog izraza. Može biti:

1. Prost iskaz (iskaz dodele vrednosti).

Primer: × =1.5 Sa desne strane jednakosti može da se nađe bilo koji prost izraz.

Napomena: Iako je u bejziku sintaksički ovaj primer potpuno jednak sa primerom za izraz, postoji suštinska razlika. Kod izraza znak = je relacioni operator, dok je kod iskaza to znak za dodelu vrednosti (tj. × ⟵ 1.5)

2. Složen iskaz (sekvenca, uslovni iskaz, iteracija itd.)

8.3.1 Bejzik interpreter Komodora

Posle inicijalizacije sistema, ulazi se u interpretersku petlju. Ona intenzivno koristi Kernal-ov editor pri pisanju programa. Ukoliko napisani iskaz počinje brojem, on se stavlja u memoriju. U suprotnom se odmah izvršava. Interpreter vodi računa da se programske linije u memoriji ažuriraju po rastućim rednim brojevima, a omogućuje i brisanje linija.

Ispravno napisane službene reči (naredbe, funkcije itd.) pretvaraju se u tokene od po jednog bajta. Ukoliko su reči nepravilno napisane, smeštaju se cele u memoriju (kao string). Na taj način se, koristeći interpreter, mogu sastavljati programi i u nekom drugom programskom jeziku ili bilo kakvi drugi tekstovi. Pri tome svaka linija mora imati svoj broj.

Posle startovanja bejzik programa (RUN), sintaksički se analizira tekst, a zatim se izvršavaju naredbe u linijama (jedna linija može sadržati više iskaza razdvojenih separatorom " : "). Sintakstička analiza se obavlja na osnovu „gramatike" osnovne verzije Microsoft-ovog bejzika.

Izvršni deo interpretera sastoji se od tri velike celine:

1. Priprema iskaza za izvršenje.
2. Izračunavanje izraza.
3. Mašinski potprogrami za izvršavanje izkaza.

Rutina za čitanje iskaza (CHRGET) uzima jedan po jedan karakter iz programske linije. Ukoliko je on token službene reči, na osnovu dispečarske tabele određuje se adresa odgovarajućeg potprograma. Ukoliko nije token, smatra se da je identifikator ili konstanta (ako je broj) pa se vrši ažuriranje tabele simbola (videti sistemske promenljive).

Ako iskaz sadrži u sebi neki izraz, potrebno je prethodno izračunati njegovu vrednost. Iskazi se pišu u takozvanoj infiks notaciji. To znači da se znaci operatora stavljaju između

operanada (npr. A+B). Pored infiks notacije postoji prefiks (+AB) I postfiks (AB+) notacija, ali one nisu tako bliske standardnom načinu pisanja.

Infiks notacija, međutim, zahteva razdvajanje operatora i vrednosti indentifikatora (operanada) u dva različita softverski generisana steka. Operacije se obavljaju između dve vrednosti na vrhu operand steka. Rezultat se vraća nazad u operand stek. Koja će se operacija prvo obaviti zavisi od prioriteta operatora (implicitno nalazi zapisan u dispečerskoj tabeli operatora) kao i od toga da li postoje zagrade.

Po nailasku na desnu zagradu, operator na vrhu steka obavlja svoju funkciju, a zatim se skida sa steka. Sledeći operator je sada na vrhu steka itd. sve do nailaska leve zagrade. Posle toga se par zagrada uklanja.

Primer: Računanje izraza (X+Y)*Z+X pri čemu su vrednosti promenljivih: X=1.5, Y=2.5 i Z=2.

red. broj	stek operatora	token	stek vred. operanda
1	prazan	(	prazan
2	(	×	prazan
3	(	+	1.5
4	+	Y	1.5
	(		
5	+	)	2.5
	(		1.5
6	prazan	*	4
7	*	Z	4
8	*	+	2
			4
9	prazan	+	8
10	+	×	8
11	+	nema	1.5
8			8
12	prazan	nema	9.5

Karakteristična situacija je pod rednim brojem 5. Nailaskom desne zagrade obavlja se operacija sa vrha operatorskog steka (u ovom slučaju +) nad dva operanda sa vrha operand steka. To su 2.5 (vrh steka) i 1.5 (prvi ispod njega). Rezultat (4) se posle skidanja dva operanda stavlja na stek, operator + se skida sa operatorskog steka, pa pošto sledi leva zagrada, par zagrada se uklanja.

8.3.2 Organizacija bejzik interpretera

$A000 (40960)
Vektor hladnog starta. Ovaj vektor sadrži adresu $E394 (58260) gde počinje rutina reinicijalizacije bejzika.

$A002 (40962)
Vektor radnog starta (engl. warm start) bejzika. Sadrži adresu $E37B (58235) gde počinje rutina za obradu softverskog prekida — BRK.

$A004 (40964)
Na ovoj lokaciji nalaze se ASCII karakteri: „CMBASIC".

$A00C – $A051 (40972 – 41041) – STMDSP
Ova tabela sadrži vektore od kojih svaki pokazuje na adresu za jedan manju od adrese početka rutine za odgovarajuću bejzik naredbu. Naredbe su date u obliku tokena i to po rastućem redosledu.
Kada je potrebno da se izvrši naredba, rutina NEWSTT – $A7AE (42926) stavlja ovu adresu umanjenu za jedan na stek, a zatim izvršava CHRGET bezuslovnim skokom (JMP). Pošto se CHRGET završava sa RTS, pri povratku se adresa umanjena za jedan, rutine za naredbu, skida sa steka, uvećava za jedan i stavlja u programski brojač.

adresa ($)	vektor ($)	naredba	token ($)	stvarna adresa rutine
A00C	30A8	END	80	$A831 (43057)
A00E	41A7	FOR	81	$A742 (42818)
A010	1DAD	NEXT	82	$AD1E (44318)
A012	F7AB	DATA	83	$A8F8 (43256)
A014	A4AB	INPUT #	84	$ABA5 (43941)
A016	BEAB	INPUT	85	$ABBF (43967)
A018	80B0	DIM	86	$B081 (45185)
A01A	05AC	READ	87	$AC06 (44038)
A01C	A4A9	LET	88	$A9A5 (43429)
A01E	9FA8	GOTO	89	$A8A0 (43168)
A020	70A8	RUN	8A	$A871 (43121)
A022	27A9	IF	8B	$A928 (43304)
A024	1CA8	RESTORE	8C	$A81D (43037)
A026	82A8	GOSUB	8D	$A883 (43139)
A028	D1A8	RETURN	8E	$A8D2 (43218)
A02A	3AA9	REM	8F	$A93B (43323)
A02C	2EA8	STOP	90	$A82F (43055)
A02F	4AA9	ON	91	$A94B (43339)
A030	2CB8	WAIT	92	$B82D (47149)
A032	67E1	LOAD	93	$E168 (57704)
A034	55E1	SAVE	94	$E156 (57686)
A036	64E1	VERIFY	95	$E165 (57701)
A038	B2B3	DEF	96	$B3B3 (46003)
A03A	23B8	POKE	97	$B824 (47140)
A03C	7FAA	PRINT #	98	$AA80 (43648)
A03E	9FAA	PRINT	99	$AAA0 (43680)
A040	56A8	CONT	9A	$A857 (43095)
A042	9BA6	LIST	9B	$A69C (42652)
A044	5DA6	CLR	9C	$A65E (42590)
A046	85AA	CMD	9D	$AA86 (43654)
A048	29E1	SYS	9E	$E12A (57642)
A04A	BDE1	OPEN	9F	$E1BE (57790)
A04C	C6E1	CLOSE	A0	$E1C7 (57799)
A04E	7AAB	GET, GET #	A1	$AB7B (43899)
A050	41A6	NEW	A2	$A642 (42562)

$A052 – $A07F (41042 – 41087) – FUNDSP
Ova tabela sadrži vektore od kojih svaki pokazuje na adresu rutine za određenu bejzik funkciju. Funkcije su date u obliku tokena po rastućem redosledu. Funkcije se razlikuju od ostalih naredbi po tome što imaju argument u zagradi. Ovaj argument može biti i neki izraz, pa se on prvo izračunava pomoću rutine FRMEVL – $AD9E (44446).
Adresa funkcije USR se nalazi u RAM-u na lokaciji $0310 (784) pa se može menjati.

adresa ($)	vektor ($)	funkcija	token ($)	stvarna adresa rutine
A052	39BC	SGN	B4	$BC39 (48185)
A054	CCBC	INT	B5	$BCCC (48332)
A056	58BC	ABS	B6	$BC58 (48216)
A058	1003	USR	B7	promenljiva

A05A	7DB3	FRE	B8	SB37D (45949)
A05C	9EB3	POS	B9	SB39E (45982)
A05E	71BF	SOR	BA	SBF71 (49009)
A060	97E0	RND	BB	SE097 (57495)
A062	EAB9	LOG	BC	SB9EA (47594)
A064	EDBF	EXP	BD	SBFED (49133)
A066	64E2	COS	BE	SE264 (57956)
A068	6BE2	SIN	BF	SE26B (57963)
A06A	B4E2	TAN	C0	SE2B4 (58036)
A06C	0EE3	ATN	C1	SE30E (58126)
A06E	0DB8	PEEK	C2	SB80D (47117)
A070	7CB7	LEN	C3	SB77C (46972)
A072	65B4	STRS	C4	SB465 (46181)
A074	ADB7	VAL	C5	SB7AD (47021)
A076	8BB7	ASC	C6	SB78B (46987)
A078	ECB6	CHRS	C7	SB6EC (46828)
A07A	00B7	LEFTS	C8	SB700 (46848)
A07C	2CB7	RIGHTS	C9	SB72C (46892)
A07E	37B7	MIDS	CA	SB737 (46903)

SA080 – SA09D (41088 – 41117) – OPTAB
Ova tabela sadrži vektore od kojih svaki pokazuje na adresu za jedan manju od adrese početka rutine za odgovarajuću bejzik matematičku operaciju. Za razloge ovakvog sadržaja vektora pogledati opis tabele STMDSP.

Uz svaki vektor pridružen je još jedan bajt koji označava stepen prioriteta dotične operacije. Operacije višeg prioriteta obavljaju se pre operacija nižeg prioriteta. Prioriteti operacija su sledeći:

1. izrazi unutar zagrade (najviši prioritet)
2. stepenovanje
3. negacija izraza (množenje sa –1, npr. –5 ili –A)
4. množenje i deljenje
5. sabiranje i oduzimanje
6. relacione operacije (=, <>, <, >, <=, >=)
7. NOT
8. AND
9. OR

Ukoliko su operacije istog prioriteta, izvršavaju se po redosledu nailaska.

adresa (S)	prioritet (S)	vektor (S)	operacija	token (S)	stvarna adresa rutine
SA080	79	69B8	+	AA	SB86A (47210)
SA083	79	52B8	–	AB	SB853 (47187)
SA086	7B	2ABA	*	AC	SBA2B (47659)
SA089	7B	11BB	/	AD	SBB12 (47890)
SA08C	7F	7ABF	^	AE	SBF7B (49019)
SA08F	50	E8AF	AND	AF	SAFE9 (45033)
SA092	46	E5AF	OR	BO	SAFE6 (45030)
SA095	7D	B3BF	>	B1	SBFB4 (49076)
SA098	5A	D3AE	=	B2	SAED4 (44756)
SA09B	64	15B0	<	B3	SB016 (45078)

SA09E – SA19D (41118 – 41373) – RESLST
Ova tabela sadrži kompletnu listu svih bejzik službenih reči u obliku ASCII teksta, pri čemu je bit 7 poslednjeg slova svake reči postavljen na jedinicu označavajući kraj reči. To je ekvivalentno tom slovu uz pritisnut taster SHIFT. Na primer, početak tabele bi uz korišćenje skupa karaktera mala/velika slova izgledao ovako:

	sadržaj	ASCII
SA09E	45 4E C4 46 4F D2 4E 45 58 D4	; enDfoRnexT itd.

Kada se unosi bejzik tekst, ova tabela se koristi za dobijanje tokena za odgovarajuće reči jer su reči poređane po rastućim vrednostima tokena. Pri tom, prva reč ima token S80, druga S81 itd. da bi se razlikovale od standardnih ASCII kodova karaktera. Prilikom listanja programa, ova tabela se takođe koristi. Naime, iz nje se na osnovu tokena, uzima cela reč i ispisuje na ekranu. Prvi deo tabele sadrži službene reči naredbi (tokeni od S80 do SA2). Drugi deo čine reči kojima nikada ne počinje bejzik iskaz.

reč	token (S)
TAB	A3
TO	A4
FN	A5
SPC(	A6
THEN	A7
NOT	A8
STEP	A9

Treći deo tabele čine simboli matematičkih operacija (tokeni od SAA do SB3).
Četvrti deo tabele čine reči funkcija (tokeni SB4 do SCA).
Na kraju tabele nalazi se reč GO (token SCB) koja postoji zbog kompatibilnosti sa starijim računarima i omogućuje razdvojeno pisanje naredbe GO TO što je ekvivalentno sa GOTO.

SA19E — SA327 (41374 — 41767) — ERRTAB
Ova tabela sadrži ASCII tekst svih bejzik izveštaja. Kao i kod RESLST tabele, bit 7 poslednjeg slova u izveštaju je postavljen na jedinicu označavajući kraj izveštaja. Izveštaji su nabrojani redosledom kojim su poređani u tabeli.

1. TOO MANY FILES
2. FILE OPEN
3. FILE NOT OPEN
4. FILE NOT FOUND
5. DEVICE NOT PRESENT
6. NOT INPUT FILE
7. NOT OUTPUT FILE
8. MISSING FILENAME
9. ILLEGAL DEVICE NUMBER
10. NEXT WITHOUT FOR
11. SYNTAX
12. RETURN WITHOUT GOSUB
13. OUT OF DATA
14. ILLEGAL QUANTITY
15. OVERFLOW
16. OUT OF MEMORY
17. UNDEF'D STATEMENT
18. BAD SUBSCRIPT
19. REDIM'D ARRAY
20. DIVISION BY ZERO
21. ILLEGAL DIRECT
22. TYPE MISMATCH
23. STRING TOO LONG
24. FILE DATA
25. FORMULA TOO COMPLEX
26. CAN'T CONTINUE
27. UNDEF'D FUNCTION
28. VERIFY
29. LOAD

SA328 — SA364 (41768 — 41828)
Ovo je tabela vektora od kojih svaki pokazuje na prvo slovo odgovarajućeg bejzik izveštaja.

SA365 — SA389 (41829 — 41865)
Ovo je tabela raznih poruka bejzika. Poruke u sebi uključuju i pomeranje kursora (obeleženo u uglastoj zagradi).

1. ⟨CR⟩, OK, ⟨CR⟩
2. ⟨SPACE⟩, ⟨SPACE⟩, ERROR
3. ⟨SPACE⟩, IN, ⟨SPACE⟩
4. ⟨CR⟩, ⟨LF⟩, READY., ⟨CR⟩, ⟨LF⟩
5. ⟨CR⟩, ⟨LF⟩, BREAK

SA38A (41866) — FNDFOR
Ova rutina pretražuje stek da bi našla blokove podataka od 18 bajtova koji se tu stavljaju prilikom otvaranja svake FOR petlje. Pronađene vrednosti se dodeljuju indeksima, pokazivačima (pointerima) promenljivih itd.

$A3B8 (41912) – BLTU
Ova rutina oslobađa prostor u memoriji za dodavanje nove programske linije ili promenljive.

$A3FB (41979) – GETSTK

Provera preostalog praznog prostora na steku. Ukoliko nema prostora, javlja se izveštaj OUT OF MEMORY.

$A408 (41992) – REASON
Provera preostalog prostora u memoriji. Ukoliko nema prostora, javlja se izveštaj OUT OF MEMORY.

$A435 (42037) – OMERR
Postavljanje koda izveštaja.

$A437 (42039)
Ispisuje reč ERROR u nastavku izveštaja. Takođe se koristi za ispisivanje reči BREAK.

$A474 (42100) READY
Ispisivanje poruke READY, uključivanje direktnog načina rada i ulazak u kontrolnu (interpretersku) petlju.

$A480 (42112) – MAIN
Kontrolna (interpreterska) petlja. U nju se ulazi preko RAM vektora $302 (770). U okviru ove rutine čita se linija teksta iz bafera tastature i ispituje da li poseduje broj linije. Ako broj postoji, izvršava se rutina koja stavlja liniju u memoriju. Ukoliko nema broja, iskazi u toj liniji se odmah izvršavaju.

$A49C (42140) – MAIN1
Ova rutina poziva potprograme koji učitavaju karaktere iz jedne linije teksta, tokenizuje službene reči a zatim traži u memoriji liniju sa istim brojem. Ukoliko je nađe, briše je pomerajući sve linije sa većim brojem i promenljive unazad za onoliko mesta koliko je ta linija zauzimala prostora. Zatim se dodaje nova linija. Pošto se poziva CLR rutina, vrednosti svih promenljivih se gube.

$A533 (42291) – LINKPRG
Svaka linija programskog teksta počinje pokazivačem adrese sledeće linije (link adresa). Ova rutina pretražuje svaku liniju sve do njenog kraja koji je označen bajtom 0. Zatim izračunava link adresu dodajući broj bajtova na adresu prethodno pretražene linije.

$A560 (42336) – INLIN
Ova rutina poziva standardnu Kernal rutinu CHRIN da bi učitala liniju sa ulazne jedinice, najčešće tastature. Linija se stavlja u ulazni bafer bejzik teksta, koji počinje na lokaciji $200 (512), sve do karaktera CR – $0D (13) ili dostizanja dužine od 89 karaktera. Tastatura kao ulazni uređaj može dati maksimalno 80 karaktera pre automatskog generisanja CR.

$A579 (42361) – CRUNCH
Kada je linija stavljena u bafer bejzik teksta – $200 (512), ova rutina ispituje liniju menjajući službene reči ili njihove skraćenice, koje nisu pod znakovima navoda, u odgovarajuće tokene. U ovu rutinu se ulazi indirektno preko RAM vektora $304 (772).

$A613 (42515) – FNDLIN
Ova rutina pretražuje programski tekst upoređujući brojeve programskih linija sa sadržajem u LINNUM – $14 (20). Ukoliko se neki broj podudara, vektor $5F (95) se postavlja da pokazuje na link adresu u toj liniji i indikator C se postavlja na jedinicu. U suprotnom, C se postavlja na nulu.

$A642 (42562) – NEW

$A65E (42590) – CLR

$A68E (42638) – RUNC
Ova rutina resetuje CHRGET pokazivač TXTPTP – $7A (122) tako da je sledeći bajt teksta koji interpreter uzima, prvi bajt programskog teksta.

$A69C (42652) – LIST

$A717 (42775) – QPLOP
Ovo je deo LIST rutine koja menja token u odgovarajući ASCII tekst naredbe. U ovu rutinu se ulazi preko RAM vektora $306 (774).

$A742 (42818) – FOR

SA7AE (42926) – NEWSTT
Ova rutina ispituje STOP taster, postavlja pokazatelj trenutnog broja linije i postavlja pokazatelj teksta na početak iskaza.

SA7E4 (42980) – GONE
Rutina koja na osnovu tokena izvršava iskaz. U nju se indirektno ulazi preko RAM vektora S308 (776). Svaki iskaz mora početi tokenom naredbe ili implicitnom LET naredbom. Ova rutina ispituje da li je prvi bajt ispravni token. Ukoliko jeste, koristeći tabelu STMDSP, ide na izvršavanje. U suprotnom javlja izveštaj SYNTAX ERROR.

SA81D (43037) – RESTORE
SA82F (43055) – STOP
SA831 (43057) – END
SA857 (43095) – CONT
SA871 (43121) – RUN
SA883 (43139) – GOSUB
SA8A0 (43168) – GOTO
SA8F8 (43218) – RETURN
SA8F8 (43256) – DATA

SA906 (43270) – DATAN
Ova rutina pretražuje programski tekst sve dok ne nađe kraj linije tj. bajt S0 – linijski delimiter ili dve tačke koje nisu pod znakovima navoda – delimiter iskaza.

SA928 (43323) – IF
SA93B (43323) – REM
SA94B (43339) – ON

SA96B (43371) – LINGET.
Ova rutina konvertuje ASCII decimalni broj u dvobajtni binarni broj linije.

SA9AS (43429) – LET
SAA80 (43648) – PRINT #
SAA86 (43654) – CMD
SAAA0 (43680) – PRINT

SAB7B (43899) – GET
Ova rutina je zajednička za GET i GET #

SABA5 (43941) – INPUT #
SABBF (43967) – INPUT
SAC06 (44038) – READ
SAD1E (44318) – NEXT

SAD8A (44426) – FRMNUM
Ospituje tip podatka posle obavljene FRMEVL rutine. Ukoliko se tip ne podudara sa očekivanim, generiše izveštaj TYPE MISMATCH.

SAD9E (44446) – FRMEVL
Ovo je početak grupe rutina intenzivno korišćenih u bejziku. Glavna funkcija im je čitanje ASCII teksta bejzik izraza, razdvajanje operatora i termova, ispitivanje grešaka, kombinovanje pojedinih termova izvršavanjem naznačenih operacija i dobijanje krajnje vrednosti koju koristi bejzik program. Sve ovo može da bude vrlo kompleksan zadatak jer izraz može da bude numeričkog ili string tipa i da bude sastavljen od bilo kog tipa promenljivih kao i od konstanti.
Na kraju se postavlja vrednost sistemske promenljive VALTYP S0D (13) u zavisnosti da li je rezultat numerički ili string kao i INTFLG S0E (14) u zavisnosti da li je rezultat celobrojan ili je u notaciji sa pokretnim zarezom (ako je numerički).

SAE83 (4467S) – EVAL
Ova rutina konvertuje pojedinačni aritmetički term, koji je deo izraza, iz njegovog ASCII oblika u ekvivalent u pokretnom zarezu. Ako je to konstanta, rutina postavlja VALTYP · S0D (13) kao broj, a pokazatelj teksta postavlja na prvi ASCII karakter. Zatim poziva rutinu koja konvertuje ASCII string u broj sa pokretnim zarezom.
Ukoliko je term promenljiva, vrednost promenljive se prvo uzima iz tabele promenljivih (na koju ukazuje vektor VARTAB – S2D (45)). U ovu rutinu se ulazi preko RAM vektora S30A (778)

$AEA8 (44712) – PIVAL
Vrednost konstante PI u obliku broja sa pokretnim zarezom.

$AEF1 (44785) – PARCHK
Ova rutina računa vrednost izraza u zagradi pozivajući prvo rutinu za sintaksičku analizu (da li su zagrade propisano otvorene i zatvorene), a zatim, za svaki nivo zagrade poziva rutinu FRMEVL.

$AF2B (44843) – ISVAR
Ukoliko je u izrazu prisutna i neka funkcija, ova rutina koristeći tabelu FUNDSP – $A052 (41042) postavlja adresu odgovarajuće rutine, a zatim je izvršava.

$AFE6 (45030) – OR

$AFE9 (45033) – AND

$B016 (45078) – DORE1
U ovoj rutini se obavlja poređenje (>,< ili =). Rezultat je 0 za netačan, ili –1 za tačan ishod poređenja.

$B081 (45185) – DIM

$B08B (45195) – PTRGET
Ova rutina u oblasti promenljivih traži određenu promenljivu. Ukoliko je tamo nema ona poziva NOTFNS rutinu.

$B11D (45341) – NOTFNS
Ova rutina kreira novu bejzik promenljivu. Ona pravi mesta u memoriji za sedmobajtovski deskriptor pomerajući oblast promenljivih za sedam bajtova naviše.

$B185 (45445) – FINPTR
Ova rutina stavlja adresu promenljive koja je pronađena (pomoću PTRGET) ili kreirana (pomoću NOTFNS) u pokazivač VARPNT – $47 (71).

$B194 (45460) – ARYGET
Ova rutina rezerviše pet bajtova plus dva bajta za svaku dimenziju navedenu u deskriptoru matrice.

$B1A5 (45477) – N32768
Konstanta 32768 u obliku broja sa pokretnim zarezom.

$B1AA (45482)
Konverzija broja sa pokretnim zarezom u ceo broj sa predznakom (engl. signed integer). Rezultat ostaje u A i Y registru.
Iako se ova rutina nigde ne koristi u bejziku, u nju se ulazi preko RAM vektora ADRAY1 – $3 (3).

$B1B2 (45490) – INTIDX
Ova rutina konvertuje indeks koji je u obliku broja sa pokretnim zarezom u ceo broj. Prethodno se proverava da li je broj pozitivan.

$B1BF (45503) – AYINT
Ova rutina prvo proverava da li je broj unutar FAC1 (engl. floating accumulator 1 – akumulator 1 za rad sa pokretnim zarezom) između 32767 i –32768. Ukoliko nije, generiše se greška ILLEGAL QUANTITY. Ukoliko jeste, vrši se konverzija u 16 bitni ceo broj sa predznakom. Pri tome je viši bajt na lokaciji $64 (100), a niži bajt na lokaciji $65 (101).

$B1D1 (45521) – OSARY
Ova rutina traži matricu sa određenim imenom. Ako je nađe, proverava se valjanost indeksa pa se zatim postavljaju pokazivači matrice, kao i pojedinačnog elementa u matrici. Ukoliko matrica ne postoji vrši se njeno kreiranje.

$B245 (45637) – BSERR
Prikazivanje izveštaja BAD SUBSCRIPT.

$B248 (45640) – FCERR
Prikazivanje izveštaja ILLEGAL QUANTITY.

$B34C (45900) – UMULT
Ova rutina računa veličinu višedimenzione matrice množeći njene dimenzije.

$B37D (45949) – FRE

$B391 (45969) – GIVAYF
Ova rutina konvertuje 16-bitni ceo broj čiji je viši bajt u A registru, a niži u Y registru, u ekvivalent sa

pokretnim zarezom. Rezultat se stavlja u FAC1. U ovu rutinu se ulazi preko RAM vektora ADRAY2 – $5 (5).

$B39E (45982) – POS

$B3A6 (45990) – ERRDIR
Ovu rutinu pozivaju one rutine koje se nikada ne izvršavaju u direktnom načinu rada. Ona proverava da li je direktan način rada aktivan, pa ako jeste, daje izveštaj ILLEGAL DIRECT.

$B3B3 (46003) – DEF

$B3F4 (46068) – FN
$B465 (46181) – STR$

$B4B7 (46215) – STRLIT
Ova rutina računa dužinu stringa pa na osnovu toga obezbeđuje prostor u memoriji.

$B526 (46374) – GARBAG
Uvek kada se na bilo koji način menja neki string, novodobijeni string se stavlja na dno oblasti za stringove. Stari string ostaje na lokacijama iznad novog pri čemu se nepotrebno zauzima memorija. Ova rutina oslobađa ovako zauzetu memoriju.

$B63D (46653) – CAT
Spajanje dva stringa (na primer A$+B$).

$B67A (46714) – MOVINS
Rutina za premeštanje stringa.

$B6EC (46828) – CHR$
$B700 (46848) – LEFT$
$B72C (46892) – RIGHT$
$B737 (46903) – MID$

$B761 (46945) – PREAM
Ova rutina prihvata parametre za LEFT$, i MID$.

$B77C (46972) – LEN
$B78B (46987) – ASC

$B79B (47003) – GETBYT
Ova rutina čita parametar u obliku ASCII teksta, pretvara ga u jednobajtovsku brojnu vrednost koju stavlja u X registar. Prethodno je provereno da li je u opsegu od 0 do 255.

$B7AD (47021) – VAL
$B7EB (47083) – GETNUM

Ova rutina čita parametar u obliku ASCII teksta, proverava da li je u opsegu od 0 do 65535, a zatim ga pretvara u dvobajtnu adresu koju smešta u LINNUM $14 (20). Nakon toga se ispituje da li postoji zarez pa se iza njega učitava jednobajtni parametar i smešta u X registar.
Ova rutina se koristi za prenos parametara kod POKE i WAIT.

$B7F7 (47095) – GETADR
Konverzija parametra u dvobajtnu adresu. Koristi se kod PEEK.

$B80D (47117) – PEEK
$B824 (47140) – POKE
$B82D (47149) – WAIT

$B849 (47177) – FADDH
FAC1 = FAC1 + 0.5. Uvećanje sadržaja float akumulatora 1 (FAC1) za 0.5.

$B850 (47184) – FSUB
FAC2 = MEM(A/Y). Prenos argumenta iz memorije, adresirane sa A (inži bajt) i Y (viši bajt) u FAC2
Zatim sledi poziv FSUBT.

$B853 (47187) – FSUBT
FAC1 = FAC1 – FAC2

$B867 (47207) – FADDT
FAC1 = FAC1 + FAC2

$B8A7 (47271) -- FADD4
Ukoliko je došlo do pozajmice, rezultat se postavlja kao negativan.

$B8FE (47358) – NORMAL
Normalizacija FAC1.

$B947 (47431) – NEGFAC
FAC1 = –FAC1

$B9BC (47548) – FONE
Konstanta 1 u obliku sa pokretnim zarezom.

$B9C1 (47553) – LOGCN2
Tabela konstanti potrebnih za izračunavanje logaritama.

$B9EA (47594) – LOG

$BA28 (47656) – FMUL
Množenje FAC1 sa sadržajem u memoriji na koju pokazuju A (niži bajt) i Y (viši bajt) registri. Prethodno se taj sadržaj prenosi u FAC2 pa se poziva FMULT.

$BA33 (47667) – FMULT
FAC1 = FAC2 * FAC1

$BA8C (47756) – CONUPK
Ova rutina puni FAC2 sadržajem četiri uzastopne lokacije u memoriji (tri bajta mantise i jedan za znak). Prva lokacija je adresirana sadržajem A i Y registara.

$BAE2 (47842) – MUL10
FAC1 = FAC1 * 10

$BAF9 (47865) – TENC
Konstanta 10 u obliku sa pokretnim zarezom.

$BAFE (47890) – DIV10
FAC1 = FAC1/10

$BB0F (47887) – FDIV
Deljenje sadržaja memorije na koji pokazuju A (niži) i Y (viši bajt) registri sa FAC1. Prethodno se taj sadržaj prenosi u FAC2 pa se poziva FDIVT.

$BB12 ($# 890) – FDIVT
FAC1 = FAC2/FAC1. Prethodno se ispituje da li je FAC1 = 0.

$BBA2 (48034) – MOVFM
Premeštanje petobajtnog broja sa pokretnim zarezom iz memorije (A/Y) u FAC1.

$BBC7 (48071) -- MOV2F
Premeštanje broja iz FAC1 u memoriju na $5C (92) ili $57 (87), zavisno od ulazne adrese u rutinu.

$BBFC (48124) – MOVFA
Kopiranje FAC2 u FAC1.

$BC0C (48140) – MOVAF
Zaokruživanje, a zatim i kopiranje FAC1 u FAC2.

$BC0F (48143) – MOVEF
Kopiranje FAC1 u FAC2 bez zaokruživanja.

$BC1B (48155) – ROUND
Zaokruživanje broja unutar FAC1.

$BC2B (48171) – SIGN
Stavljanje znaka akumulatora FAC1 u A registar. Ako je FAC1 = 0 u A se stavlja 0, ako je pozitivan stavlja se 1, a ako je negativan stavlja se $FF (255).

$BC39 (48185) – SGN

$BC58 (48216) – ABS

$BC5B (48219) – FCOMP
Registri A i Y pokazuju na memorijsku lokaciju gde je smešten petobajtni broj sa pokretnim zarezom koji se poredi sa sadržajem FAC1. Ako su jednaki, u A ide 0. Ako je broj u FAC1 veći, u A ide 1, a ako je manji A = $FF.

$BC9B (48283) – QINT
Konverzija broja unutar FAC1 u četvorobajtni ceo broj sa predznakom. Najteži bajt je na $62 (98), a najlakši na $65 (101).

$BCCC (48332) – INT

$BCF3 (48371) – FIN
Konverzija ASCII stringa u broj sa pokretnim zarezom i smeštanje u FAC1.

$BDDD (48605) – FOUT
Konverzija broja u FAC1 u ASCII string i postavljanje pokazivača (A/Y) na početak stringa.

$BF11 (48913) – FHALF
Konstanta 0.5 u obliku sa pokretnim zarezom.

$BF71 (49009) – SQR
$BF7B (49019) – FPWRT
FAC1 = FAC2↑FAC1
$BFB4 (49076) – NOT
$BFED (49133) – EXP
$BFFD (49149) – JMP $E000
Skok u Kernal ROM.

$E000 (57344)
Nastavak EXP rutine.

$E097 (57495) – RND
$E12A (57642) – SYS
$E156 (57686) – SAVE
$E165 (57710) – VERYFY
$E168 (57704) – LOAD
$E1BE (57790) – OPEN
$E1C7 (57799) – CLOSE
$E264 (57956) – COS
$E268 (57960) – SIN
$E2B4 (58036) – TAN
$E2E0 – $E30D (58060 – 58125) – Tabele raznih konstanti.

$E30E (58126) – ATN

$E37B (58235) – WARM
Ovo je radni početak bejzika u koji se ulazi iz BRK rutine $FE66 (65126), a koja se izvršava kada su pritisnuti istovremeno STOP i RESTORE tasteri. Prvo se poziva Kernal CLRCHN rutina koja zatvara sve datoteke i kanale. Zatim se postavljaju standardni ulazno/izlazni uređaji (tastatura, ekran) pa se vrši indirektan skok na sledeću rutinu preko RAM vektora $300 (768).

$E38B (58251) – ERRMH
Ova rutina prikazuje izveštaj iz tabele ERRTAB – $A193 (41363) indeksirano sa X ili prikazuje poruku READY ako nema izveštaja, a zatim preko RAM vektora $302 (770) ide u interpretersku petlju.

$E394 (58260) – COLD
Ova rutina se izvršava uvek po izvršenom resetovanju sistema. Ona postavlja RAM vektore počev od $300 (768), inicijalizuje interpreter, prikazuje reči početne poruke i na kraju ulazi u interpretersku petlju.

$E3A2 (58274) – INITIAT
Kopija CHRGET rutine koja se prebacuje u RAM počev od $73 (115) za vreme inicijalizacije bejzika

$E3BF (58303) – INIT
Inicijalizaciona rutina bejzika. Ona kopira CHRGET u RAM.

$E422 (58402) PRSUM
Rutina koja prikazuje početnu poruku.

$E447 (58439) — RAMTBL
Tabela vektora važnih bejzik rutina i tabela. Ova se tabela kopira u RAM počev od $300 (768).

$E453 (58451) — VCOPY
Ova rutina kopira prethodnu tabelu u RAM.

$E460 (58464) — WORDS
ASCII tekst početne poruke: „*** COMMODORE 64 BASIC V2 ****" i „BYTES FREE".

Primer: Poboljšani način računanja funkcije SQR(x) primenom Heronovog obrasca.

U Komodoru se kvadratni koren računa pomoću formule Y = EXP (0.5*LOG(x)). Ovo računanje obično ne može da se prihvati jer je SQR(x) najčešća funkcija koja mora da se izračuna tačno i brzo. Čak i pod pretpostavkom da su funkcije EXP i LOG besprekorno urađene, za male pozitivne i velike vrednosti argumenta ne dobija se tačna vrednost. Osim male brzine izvršavanja mogu biti netačna i do sedam bita mantise.

Sledeći primer pokazuje primenu Heronovog obrasca Yn = (X/Yn − 1 + Yn − 1)/2. Da bi se smanjio broj iteracija treba pravilno odrediti početnu vrednost Yo. Ukoliko se upotrebi aproksimacija Yo = m*2 ↑INT((k + 1)/2), gde je m — mantisa, a k — eksponent (karakteristika), dobija se sa samo 5 iteracija tačnost od najmanje 32 bita (10 značajnih cifara).

Da bi se u računar unela izmena, prvo je potrebno kopirati bejzik interpreter u RAM, a zatim promeniti vektor SQR rutine. Kod poziva ove funkcije, argument se nalazi u akumulatoru za rad sa pokretnim zarezom FAC1, pri čemu je eksponent u lokaciji FACEXP. Eksponent može biti i negativan, pa o ovome treba voditi računa.

```
10 REM *** KUADRATNI KOREN ***
20 REM ***********************
30 :
100 SYS 8*4096
101 .OPT OO
102 *- $C000
103 :SQR - $AC5E
104 :COUNT - $FB
105 ;
106 ; KOPIRANJE INTERPRETERA U RAM
107 ; --------------------------
108 :         LOY #$0C
109 :         STY COUNT
110 :         LDA #$A0
111 :         STA COUNT+1
112 :NEXT     LOA (COUNT),Y
113 :         STA (COUNT),Y
114 :         INY
115 :         BNE NEXT
116 :         INC COUNT+1
117 :         LOA COUNT+1
118 :         CMP #$C0
119 :         BNE NEXT
120 ;
121 ; UKLJUCIVANJE LORAM-A
122 ; ------------------
123 :         LOA $01
124 :         ANO #$FE
125 :         STA $01    ; LORAM ON
126 ;
127 ; IZMENA VEKTORA SQR RUTINE
128 ; ------------------------
129 :         LOA #<KOREN
130 :         STA SQR
131 :         LDA #>KOREN
132 :         STA SQR+1
133 :         RTS
134 KOREN - *
135 FACEXP - $61
136 MOV1F - $BBCA
137 MOV2F - $BBC7
138 ARG1 - $5C
139 ARG2 - $57
140 MOVFM - $BBA2
141 FDIV - $BB0F
142 FMUL - $BA28
143 FADD - $B867
144 FHALF - $BF11
145 :         JSR MOV2F   , ARG1-FAC1
146 :         BEQ IZLAZ
147 ;
148 ; ODREDJIVANJE POCETNE VREDNOSTI
149 ; ----------------------------
150 :         LDA FACEXP
151 :         CMP #128   ; DA LI JE EKSPONENT VECI OD NULE
152 :         BMI NEG
153 :         SEC
```

```
 :        SBC #127  ; EKSP.=EKSP.-128+1
 :        LSR A     ; EKSP.=EKSP./2
 :        ADC #128  ; EKSP.=EKSP.+128
 :        STA FACEXP
 :        JMP BEGIN
 :NEG     LDA #128  ; SLUCAJ NEGATIVNOG EKSPONENTA
 :        SEC
 :        SBC FACEXP; EKSP.=129-EKSP.
 :        STA FACEXP
 :        LSR FACEXP; EKSP.=EKSP./2
 :        LDA #129
 :        CLC
 :        SBC FACEXP; EKSP.=128+EKSP.+1
 :        STA FACEXP
 :BEGIN   JSR MOV1F ; ARG2=POC.VRED.
 :        LDA #<ARG2
 :        LDY #>ARG2
 :        JSR MOVFM ; FAC1=ARG2
 :        LDA #5    ; BROJ ITERACIJA
 :        STA BRDJAC
 ;
 ; RACUNANJE IZRAZA Y=(X/Y+Y)*0.5
 ; ------------------------------
 ;
 :LOOP    LDA #<ARG1
 :        LDY #>ARG1
 :        JSR FDIV ; FAC1=ARG1/FAC1
 :        LDA #<ARG2
 :        LDY #>ARG2
 :        JSR FADD  ; FAC1=FAC1+ARG2
 :        LDA #<FHALF
 :        LDY #>FHALF
 :        JSR FMUL  ; FAC1=FAC1*0.5
 :        JSR MOV1F ; ARG2=FAC1
 :        DEC BROJAC
 :        BNE LOOP
 :IZLAZ   RTS
 :BROJAC  *=.*
 .END
```

8.4 OPERATIVNI SISTEM

Operativni sistemi su sistemski programi koji nadgledaju i upravljaju radom računara. Oni olakšavaju pisanje i razvoj programa, smanjujući broj naredbi koje treba da napišu korisnici.

Operativni sistemi mogu biti:

1. Za rad u realnom vremenu (engl. real time)
2. Za sukcesivan pristup računaru (engl. batch)
3. Za paralelni pristup računaru (engl. time sharing)
4. Za višenamenski rad (engl. multipurpose)

Funkcija operativnih sistema kod velikih računara je višestruka. On omogućuje paralelno izvršavanje više programa (engl. multitasking), korišćenje računara od strane više korisnika (engl. multiuser), štiti programe i podatke itd. Operativni sistem čine: jezgro, korisnički programi (engl. utility), programi za prevođenje (engl. compiler), programi za povezivanje (engl. linker) i uređivanje (engl. editor).

Jezgro operativnog sistema (engl. Kernel) se sastoji od skupa programa koji se inten-

zivno koriste od strane drugih programa. Ono prividno predstavlja proširenje skupa naredbi hardvera na kome je operativni sistem instaliran i rezidentno je u memoriji. Funkcije jezgra u opštem slučaju su:

1. Obrada prekida. Ovo je karakteristična operacija vezana za poziv programa koji nadgleda rad sistema (engl. supervisor call), obradu zahteva za ulazom/izlazom, ispitivanje programa, ažuriranje časovnika itd.

2. Raspoređivanje. To je odlučivanje koji će program (kod multitasking sistema) da bude sledeći izvršen i na bazi toga pripremanje njegove adrese.

3. Obrada ulaza i izlaza. Informacije koje su potrebne za obavljanje ovih operacija su:
- adresa uređaja
- funkcija koja treba da se obavi
- adresa podataka

Programi (rutine) za obradu ulaza i izlaza koriste se kompleksnim strukturama podataka kao što su tabele, liste parametara itd. Ukoliko u pozivanju ovih rutina nema grešaka formiraju se sledeće strukture:

a) Tabela dodele programa i odgovarajućeg zahteva za ulaz/izlaz (engl. task control block). Ovo važi samo za multitasking operativne sisteme.

b) Tabela dodele zahteva odgovarajućem uređaju (engl. unit control block)

c) Tabela dodele zahteva kanalu (engl. input/output block)

d) Prostor u memoriji uređaja namenjen podacima (engl. data extent block)

Kanal (kanalski program) je program koji omogućuje komunikaciju između računara i periferijskih jedinica. On je skup odgovarajućih mašinskih rutina (engl. drivers).

Rutina za obradu ulaza i izlaza na osnovu tabele b) ispituje da li je periferijski uređaj dostupan. Uređaj može biti nedostupan u slučaju da je zauzet, da je njegov kontroler zauzet, ili da je na primer kod disk jedinice glava u fazi traženja odgovarajuće staze. Ukoliko je uređaj dostupan, ispituje se na osnovu tabele c) da li su odgovarajući kanali otvoreni, i ako jesu da li su slobodni. Otvaranje kanala je posao koji treba da obavi aplikacioni ili drugi sistemski program koji poziva ovu rutinu.

Kada su i uređaj i kanal slobodni, počinje ulazno/izlazna operacija koristeći tabelu d).

4. Upravljanje memorijom. Često je memorija fizički veća od adresnog prostora koji može adresirati centralna procesorska jedinica. Deo operativnog sistema koji služi za upravljanje memorijom određuje kada će koji deo memorije biti aktiviran. Kod multitasking sistema, upravljanje memorijom podrazumeva i dodeljivanje dela memorije odgovarajućem programu u odgovarajućem trenutku.

Operativni sistem Komodora — Kernal

U Komodoru se nalazi operativni sistem — Kernal koji je sveden samo na najosnovnije operacije jezgra operativnog sistema. To su operacije inicijalizacije, obrade prekida kao i ulazno izlazne operacije. Deo sistema za kontrolu disk jedinice (engl. disk operating system — DOS) nalazi se u ROM-u same disk jedinice. Kao deo Kernala se smatra i program za uređenje programskog teksta — editor.

— Prekidi

Postoje dve vrste prekida koje Kernal obrađuje.

1) IRQ. Vrši se očitavanje tastature i ažuriranje časovnika realnog vremena kao i ostalih funkcija vezanih za realno vreme.

2) NMI. Obavlja se komunikacija preko RS 232 veze kao i radni (engl. warm) start računara tj. bez prethodne reinicijalizacije sistema.

– Ulazno izlazne operacije

Ovaj deo sistema sastavljan je od celina organizovanih u potprograme za obavljanje određenih elementarnih zadataka: inicijalizacija, slanje podataka na periferni uređaj, primanje podataka sa perifernog uređaja i slično.

Operativni sistem je podložan promenama i usavršavanju. To dovodi do neminovnih izmena adresa pojedinih programa (rutina) iz kojih je on sastavljen. Iako se rutine mogu pozivati direktno iz memorije preko svojih apsolutnih adresa, taj metod se ne primenjuje jer ne bi funkcionisao kod starijih ili budućih varijanti Kernala. Da bi se ovo izbeglo primenjuju se indirektni pozivi preko takozvane tabele skokova (engl. jump table). Ova tabela se kod svih varijanti Kernala nalazi na istoj adresi: $FF81 (65409) i sadrži adrese (tzv. vektore) odgovarajućih rutina. 15 elemenata tabele u opsegu adresa $FFC0 – $FFFA (65472 – 65514) su identični za sve Komodorove računare uključujući i najstarije varijante PET-a. Ukoliko se promeni adresa neke rutine, nova vrednost se unosi u njen vektor u tabeli skokova. Pri tome adresa vektora ostaje nepromenjena.

U daljem tekstu je opisana svaka rutina pojedinačno. Date su i njihove apsolutne adrese i adrese u tabeli skokova. Pojedine rutine ne mogu da funkcionišu ukoliko se prethodno ne pozovu neke druge, pripremne, rutine. Zbog toga su i one navedene u opisu.

8.4.1 Dokumentovane rutine i upotreba

Inicijalizacione rutine

AOINT

Namena: Provera prisustva autostart ROM-a
Apsolutna adresa: $FD02 (64770)
Registri za komunikaciju: P: Ako je Z=1, postoji zaglavlje
Registri koji se koriste: A, X

Ova rutina proverava prisustvo autostart ROM-a na adresi $8000 (32768). Poredi se sadržaj zaglavlja na adresi $8004 (32772) sa sadržajem teksta na adresi $FD10 (64784)

$FD10 C3C2CD3830; „CBM80"

Ukoliko je ovaj tekst prisutan, postavlja se pokazatelj Z na jedinicu. U suprotnom se Z postavlja na nulu.

IOINIT

Namena: Inicijalizacija perifernih jedinica
Apsolutna adresa: $FDA3 (64931)
Adresa u tabeli skokova: $FF84 (65412)
Registri koji se koriste: A, X, Y

Inicijalizacija CIA 1, CIA 2 i postavljanje jačine zvuka SID integrisanog kola na nulu. Kao deo ove inicijalizacije se postavlja i tajmer A kola CIA 1 za generisanje IRQ prekida svakih 1/60 sekunde i to na osnovu informacije o TV sistemu (u Jugoslaviji je PAL sistem).
Kapija unutar mikroprocesora se postavlja na sledeći način:
– Bit 0–3 i bit 5 su izlazni – adrese memorijskih segmenata, kontrole kasetofona
– Bit 4 ulazni – ulaz sa tastera PLAY na kasetofonu.

RAMTAS

Namena: Inicijalizacija i testiranje memorije
Apsolutna adresa: $FD50 (64848)
Adresa u tabeli skokova: $FF87 (65415)
Registri koji se koriste: A, X, Y

U memorijske lokacije na nultoj, drugoj i trećoj strani upisuju se nule. Pokazivač kasetnog bafera se postavlja tako da pokazuje adresu $33C (828). To je početna adresa, dok je sledećih 192 lokacije rezervisano za rad sa kasetofonom. Posle ovoga se obavlja nedestruktivno testiranje RAM-a počev od lokacije $400 (1024). Sadržaj svake lokacije se privremeno čuva u X registru, a u nju se upisuje binarni broj 01010101. Zatim se sa iste lokacije vrši čitanje i upoređivanje sa prethodno navedenim brojem. Isto se ponavlja i za binarni broj 10101010. Na kraju se vrednost iz X registra vraća u memoriju. Kada

rezultat testiranja postane negativan (nailazak na ROM), adresa poslednje RAM lokacije stavlja se u MEMSIZ – $282 (643) označavajući kraj neprekidnog RAM-a.
Lokacije od $C000 – $CFFF se ne testiraju. Pokazivač ekranske memorije, HIBASE – $288 (648) postavlja se da pokazuje na $400 (1024), a pokazivač MEMSTR – $281 (641) na $800 (2048) tj. početak korisničkog RAM-a.

RESTOR
Namena: Inicijalizacija vektora
Apsolutna adresa: $FD15 (64789)
Adresa u tabeli skokova: $FF8A (65418)
Registri koji se koriste: A, X, Y
Potrebe za stekom: 2

Iz tabele se počev od $FD30 (64816) kopira 16 ulazno-izlaznih vektora i vektora prekida u tabelu počev od $314 (788) koristeći rutinu VECTOR.

VECTOR
Namena: Pristupanje vektorima
Apsolutna adresa: $FD1A (64794)
Adresa u tabeli skokova: $FF8D (65421)
Registri za komunikaciju: X:niži bajt adrese, Y:viši bajt adrese,
P:C =0 postavljanje, C =1 čitanje
Registri koji se koriste: A, X, Y
Potrebe za stekom: 2

Ukoliko je pri pozivu ove rutine indikator C postavljen na jedinicu, tabela će počev od adrese $314 (788) biti kopirana u tabelu počev od lokacije na koju ukazuju registri X i Y. Ukoliko je indikator C postavljen na nulu, tabela na koju ukazuju X i Y registri biće kopirana u tabelu počev od $314 (788). Iako se na ovaj način mogu menjati vektori rutina za obradu prekida, prekidi su omogućeni. IRQ treba onemogućiti pre poziva ove (ili RESTOR) rutine, a omogućiti ga po izlasku iz nje.

CINT
Namena: Inicijalizacija VIC-a i ekranskog editora
Apsolutna adresa: $FF5B (65371)
Adresa u tabeli skokova: $FF81 (65409)
Registri koji se koriste: A, X, Y
Potrebe za stekom: 4

Prvo se tastatura postavlja kao ulazni, a ekran kao izlazni uređaj. Zatim se iz tabele sa adrese $ECB9 (60601) prepisuju podaci u registre VIC-a. Posle ovoga dolazi inicijalizacija treptanja kursora, postavlja se adresa rutine za dekodovanje tastature, brzina ponavljanja pritisnutog tastera, trenutna boja karaktera i maksimalna veličina bafera tastature. Slede rutine za postavljanje link tabele na $D9 (217), za brisanje ekrana i za postavljanje kolor RAM-a na boju pozadine.
U PNTR – $D3 (211) i TBLX – $D6 (214) se upisuju nule, a pokazivač PNT – $D1 (209) se postavlja na adresu prvog bajta logičke linije uzimajući u obzir stanje ekranske link tabele, koja može ukazivati na to da dve fizičke linije treba da budu spojene u jednu logičku liniju.
Na kraju sledi provera da li je raster registar VIC-a inicijalizovan na $137 (311). Ukoliko jeste, PAL registar na adresi $2A (678) postavlja se na vrednost za PAL sistem. Tajmer A u CIA i na bazi ovoga generiše IRQ svakih 1/60 sekunde, deleći frekvenciju sistemskog takta sa vrednošću iz PAL registra.

RESET

Namena: Hladni start računara.
Apsolutna adresa: $FCE2 (64738)
Adresa u tabeli skokova: $FFFC (65532)
Registri koji se koriste: A, X, Y

Po uključenju računara, obavlja se automatski hardverski reset pri čemu se koriste inicijalizacione rutine.

Primer:
```
5 REM ****HLADNI START RACUNARA****
10 SYS (8*4096)
20 .OPT OO
30 *= $7000
40 ADINT = $FD02
```

```
50 IOINIT - $FDA3
60 RAMTAS - $FD50
70 RESTOR - $FD15
80 CINT - $FF5B
100 :RESET  LDX #$FF
110 :       SEI
120 :       TXS
130 :       CLD
140 :       JSR AOINT
150 :       BNE SKIP
160 :       JMP ($8000)
170 :SKIP   STX $D016
180 :       JSR IOINIT
190 :       JSR RAMTAS
200 :       JSR RESTOR
210 :       JSR CINT
220 :       CLI
230 :       JMP ($A000)
240 .END
```

SETMSG

Namena: Kontrola poruka operativnog sistema.
Apsolutna adresa: $FE18 (65048)
Adresa u tabeli skokova: $FF90 (65424)
Registri za komunikaciju: A: pri pozivu – broj koji ide u MSFLG, pri povratku STATUS
Registri koji se koriste: A
Potrebe za stekom: 2

Ova rutina postavlja vrednost sistemske promenljive MSGFLG – $9D (157). Na ovaj način se kontroliše prikazivanje poruka operativnog sistema.

sadržaj MSGFLG	funkcija
$C0 (192)	prikazivanje poruka o greškama i kontrolnih poruka
$80 (128)	prikazivanje kontrolnih poruka
$40 (64)	prikazivanje poruka o greškama
$0 (0)	sprečavanje prikazivanja poruka

Kontrolne poruke su tipa SEARCHING FOR, LOADING, SAVING i slično. Poruke o greškama se nikada neće javiti pri radu u bejziku jer bejzik pri izvršavanju onemogućuje njihovo ispisivanje. U bejziku se ispisuju mnogo jasnije poruke kao što je na primer FILE NOT FOUND ERROR.
Poruke Kernala o greškama su tipa I/O ERROR=nn, gde je nn broj greške sa sledećim značenjima:

broj	značenje
0	Rutina završena pomoću STOP tastera.
1	Suviše otvorenih datoteka.
2	Datoteka je već otvorena.
3	Datoteka nije otvorena.
4	Datoteka sa datim imenom nije pronađena.
5	Uređaj nije priključen.
6	Datoteka nije ulazna.
7	Datoteka nije izlazna.
8	Nedostaje ime datoteke.
9	Broj ulazno/izlaznog uređaja je nepravilan.
240	Operativni sistem je sam odredio prostor za RS232 bafer. Ovo nije greška, već upozorenje.

Ako za vreme izvršavanja neke Kernal rutine dođe do greške, indikator C se postavlja na jedinicu dok se broj greške nalazi u akumulatoru. Treba napomenuti da neke ulazno/izlazne rutine ne koriste navedene kodove, već se do njih može doći korišćenjem READST rutine.

MEMTOP

Namena: Čitanje ili postavljanje pokazivača najviše slobodne lokacije RAM-a.
Apsolutna adresa: $FE25 (65061)

Adresa u tabeli skokova: SFF99 (65433)
Registri za komunikaciju: X: niži bajt adrese. Y: viši bajt adrese. P:C = 1 čitanje. C = 0 postavljanje
Registri koji se koriste: X, Y
Potrebe za stekom: 2

Pri pozivu se vrši čitanje ako je C indikator na jedinici.
Ova rutina se poziva pri inicijalizaciji sistema, posle RAM testa. Posle inicijalizacije bejzika, ovom rutinom nije više moguće ograničiti prostor za bejzik programe.

Primer: Premeštanje RS232 bafera za 256 bajtova niže.

```
10 SYS 8*4096
20 MEMTOP = $FF99
30 .OPT 00
40 ;
100 :        SEC
110 :        JSR MEMTOP   ; CITANJE
120 :        DEY
130 :        CLC
140 :        JSR MEMTOP   ; POSTAVLJANJE
150 .END
```

MEMBOT

Namena: Čitanje i postavljanje pokazivača prve slobodne korisničke lokacije RAM-a.
adresa: SFE34 (65076)
Adresa u tabeli skokova: SFE9C (65436)
Registri za komunikaciju: X: niži bajt adrese. Y: viši bajt adrese P:C =1 čitanje. C = 0 postavljanje
Registri koji se koriste: X, Y

Pri pozivu se vrši čitanje ako je C pokazatelj postavljen na jedinicu. U suprotnom se vrši postavljanje. Adresa se nalazi u registrima X i Y.

Primer: Pomeranje donje granice korisničkog RAM-a za jednu stranicu.

```
10 SYS 8*4096
20 MEMBOT = $FF9C
30 .OPT 00
40 ;
100 :        SEC
110 :        JSR MEMBOT   ; CITANJE
120 :        INY          ; POVECANJE ADRESE ZA 256
130 :        CLC
140 :        JSR MEMBOT   ; POSTAVLJANJE NOVE VREDNOSTI
150 .END
```

IOBASE

Namena: Čitanje bazne adrese registara ulazno/izlaznih jedinica
Apsolutna adresa: SE500 (58624)
Adresa u tabeli skokova: SFFF3 (65523)
Registri za komunikaciju: X: niži bajt adrese. Y: viši bajt adrese
Registri koji se koriste: X, Y
Potrebe za stekom: 2

Pročitana adresa se nalazi u registrima X i Y. Ona omogućuje korisniku da postavi pokazivač u nultoj strani memorije i da zatim čita ili upisuje indirektno preko tog pokazivača. Na taj način se može pristupiti registrima pojedinih jedinica bez obzira gde su one fizički smeštene u memorijskom prostoru. Ovo je značajno za korišćenje istog softvera u budućim, eventualno izmenjenim, Komodorovim modelima.
U sadašnjoj verziji, ova rutina puni X registar sa S00, a Y sa SDC pokazujući na nulti registar jedinice CIA1 i na adresi SDC00.

Primer: Postavljanje korisničkog priključka da radi kao ulazni. Pretpostavka je da su CIA 1 i CIA 2 smeštene u dve susedne stranice memorije.

```
5 REM **** PRIMER ZA IOBASE ****
10 SYS(8*4096)
20 .OPT 00
30 *= $7000
```

```
40 IOBASE = $FFF3
50 POINT = $009B
90 ;
100 :     JSR IOBASE;ADRESA CIA#1
110 :     INY ;BAZNA ADRESA CIA#2
120 :     STX POINT
130 :     STY POINT+1
140 :     LDY #3;OFSET ADRESA ZA DDR
150 :     LDA #0;BROJ KOJI IDE U DDR
160 :     STA (POINT),Y;DDR=0,SVI IZVODI SU ULAZNI
170 RTS
180 .END
```

SCREEN
Namena: Čitanje formata ekrana
Apsolutna adresa: SE505 (58629)
Adresa u tabeli skokova: SFFED (65517)
Registri za komunikaciju: X: broj kolona, Y: broj redova
Registri koji se koriste: X, Y
Potrebe za stekom: 2

Omogućuje izvršavanje programa na računarima različitog formata ekrana. U sadašnjoj verziji, ova rutina puni X registar sa S28 (40), a Y sa S19 (25).

Ulazno/izlazne rutine

READST
Namena: Čitanje trenutnog statusa ili greške uređaja po obavljenoj ulazno/izlaznoj operaciji.
Apsolutna adresa: SFE07 (65031)
Adresa u tabeli skokova: SFFB7 (65463)
Registri za komunikaciju: A: status, P: Z=1 ako je status =0
Registri koji se koriste: A
Potrebe za stekom: 2

Vrednost statusa se dobija u akumulatoru sa lokacije STATUS — S90 (144) i ima sledeća značenja:
za kasetofon:

S04 (4) — kratak blok
S08 (8) — dugačak blok
S10 (16) — nepopravljiva greška u čitanju, neslaganje
S20 (32) — greška u proveri (engl. checksum error)
S40 (64) — kraj datoteke
S80 (128) — kraj trake

za uređaje sa serijskog (IEC) priključka:
S01 (1) — ispad iz sinhronizacije pri upisu (engl. time out)
S02 (2) — ispad iz sinhronizacije pri čitanju
S40 (64) — kraj ili identifikacija — EOI (engl. end or identify)
S80 (128) — uređaj nije priključen

Ukoliko se radi sa ulazno/izlaznim uređajem priključenim na RS232, status se dobija čitanjem lokacije RSSTAT — S297 (663):

S01 (1) — greška u parnosti
S02 (2) — greška u sinhronizaciji rama (engl. framing error)
S04 (4) — prepunjen bafer prijemnika
S08 (8) — prazan bafer prijemnika
S10 (16) — CTS (engl. clear to send) signal nedostaje
S20 (32) — neiskorišćen
S40 (64) — DTR (engl. data set ready) signal nedostaje
S80 (128) — prekid u komunikaciji

Treba napomenuti da se pri pozivu ove rutine, sadržaj lokacije RSSTAT briše (upisuje se nula), dok lokacija STATUS ostaje neizmenjena.

SETNAM
Namena: Postavljanje imena datoteke.
Apsolutna adresa: $FDF9 (65017)
Adresa u tabeli skokova: $FFBD (65469)

Registri za komunikaciju: X: početna adresa (niži bajt), Y: početna adresa (viši bajt), A: dužina stringa

Na osnovu sadržaja X i Y registra postavlja se pokazivač na ASCII tekst čija je dužina određena sadržajem akumulatora. Ovaj tekst je ime datoteke koja se koristi kod rutina OPEN, LOAD ili SAVE. Ako ime nije potrebno, treba pozvati rutinu sa nulom u akumulatoru.

SETLFS
Namena: Postavljanje logičke, primarne i sekundarne adrese.
Apsolutna adresa: $FE00 (65024)
Adresa u tabeli skokova: $FFBA (65466)
Registri za komunikaciju: A: logička adresa, X: primarna adresa (broj uređaja), Y: sekundarna adresa
Potrebe za stekom: 2

Ovom rutinom se definišu parametri pre pozivanja OPEN, LOAD i SAVE rutina. Logička adresa je proizvoljan broj po kojem će se datoteka identifikovati u daljem radu.

0	tastatura		nema značaja	
1	kasetofon	SAVE	0 – snimanje bez čuvanja izvorne adrese	
			1 – snimanje sa čuvanjem izvorne adrese	
			2 – po završenom snimanju staviti EOT (engl. end of tape) marker	
			3 – kombinacija 1 i 2	
		LOAD	0 – upis od početka korisničke memorije	
			1 – upis na izvornu adresu	
		OPEN	0 – čitanje datoteke	
			1 – otvaranje nove datoteke	
			2 – po završenom snimanju staviti EOT marker uvek je 0	
2	RS232			
3	video memorija		nema značaja	
4	štampač		0 – velika slova/grafički karakteri	
			7 – mala slova/velika slova, ostale vrednosti zavise od tipa štampača	
6	ploter		zavisno od plotera	
			primer za tip 1520	
			0 – štampanje ASCII karaktera	
			1 – X, Y crtanje	5 – podvlačenje
			2 – izbor boje	6 – mala/velika slova
			3 – veličina karaktera	7 – reset
			4 – rotacija karaktera	
8	disk	SAVE	uvek je 0	
		LOAD	0 – upis od početka korisničke memorije	
			1 – upis na izvornu adresu	
		OPEN	2–14 – čitanje/kreiranje datoteke	
			15 – komandni kanal	

Ako se ne koristi sekundarna adresa, u Y registru treba upisati $FF (255).

OPEN
Namena: Otvaranje logičke datoteke.
Apsolutna adresa: preko vektora $31A (794) na $F34A (62282)
Adresa u tabeli skokova: $FFC0 (65472)
Registri koji se koriste: A,X,Y

Pripremne rutine: SETLFS, SETNAM
Greške: 1,2,4,5,6,240, STATUS

Prethodno je potrebno pozvati rutine SETLFS i SETNAM. Svim sekundarnim adresama se bit 6 i 5 postavljaju na jedinicu, a zatim se logičke, primarne i sekundarne adrese upisuju u tabelu dotične datoteke čime se ona smatra otvorenom.

Primer: Ekvivalent bejzik naredbe OPEN 1,8,4, „PRIMER, SEQ, W"

```
5 REM **** OPEN PRIMER ****
10 SYS (8*4096)
20 .OPT OO
30 *= $7000
40 SETNAM = $FFBD
50 SETLFS = $FFBA
60 OPEN = $FFC0
90 ;
100 :       LDA #12
110 :       LDX #<NAME
120 :       LDY #>NAME
130 :       JSR SETNAM
140 :       LDA #1
150 :       LDX #8
160 :       LDY #4
170 :       JSR SETLFS
180 :       JSR OPEN
190 :       BCS EXIT
200 :NAME .ASC "PRIMER,SEQ,W"
210 :EXIT JMP $C000
220 :       RTS
230 .END
```

Napomena: OPEN rutina će pozvati MEMTOP i od dobijene adrese na niže odvojiti 512 bajtova za RS232 bafere. Pošto su u toj oblasti memorije najčešće smešteni bejzik stringovi, oni će biti uništeni. Zbog toga se preporučuje otvaranje datoteke pre definisanja alfanumeričkih promenljivih.
Ako se u pokazivače RS232 bafera RIBUF – $F7 (247) i ROBUF – $F9 (249) pre poziva OPEN upišu adrese koje nisu na nultoj strani, prethodni problem može se izbeći. Osim toga baferi se lociraju u proizvoljnoj oblasti memorije (na koju ukazuju pokazivači).

CHKOUT
Namena: Dodeljivanje izlaznog kanala otvorenoj datoteci.
Apsolutna adresa: preko vektora $320 (800) na $F250 (62032)
Adresa u tabeli skokova: $FFC9 (65481)
Registri za komunikaciju: X: logički broj datoteke
Registri koji se koriste: A, X
Potrebe za stekom: 4
Pripremne rutine: OPEN
Greške: 0,3,5,7, STATUS

Logički broj se prethodno unese u registar X. Po povratku iz ove rutine, u sistemskoj promneljivoj DFLTO – $9A (154) će se naći primarna adresa, to jest broj, uređaja koji je izlazni. Po uključenju računara, rutina CINT postavlja ovu vrednost na 3 tako da za ispisivanje na ekranu nije potrebno pozvati CHKOUT. Kod rada sa serijskom (IEC) vezom, u okviru ove rutine se pozivaju LISTEN i SECOND rutine.

CHKIN
Namena: Dodeljivanje ulaznog kanala otvorenoj datoteci
Apsolutna adresa: preko vektora $31E (798) na $F20E (61966)
Adresa u tabeli skokova: $FFC6 (65478)
Registri za komunikaciju: X: logički broj datoteke
Registri koji se koriste: A, X
Pripremne rutine: OPEN
Greške: 0,3,5,7, STATUS

Logički broj datoteke se prethodno unese u registar X. Po povratku iz ove rutine, u sistemskoj promenljivoj DFLTN – $99 (153) će se naći primarna adresa, to jest broj uređaja koji je ulazni. Po uključenju računara, rutina CINT postavlja ovu vrednost na 0, tako da za čitanje sa tastature nije potrebno pozivati CHKIN.
Kod rada sa serijskom (IEC) vezom, u okviru ove rutine pozivaju se TALK i TKSA rutine.

CHROUT
Namena: Slanje karaktera izlaznim kanalom
Apsolutna adresa: preko vektora $326 (806) na $F1CA (61898)
Adresa u tabeli skokova: $FFD2 (65490)
Registri za komunikaciju: A: ASCII kod karaktera
Potrebe za stekom: 8
Pripremne rutine: OPEN, CHKOUT
Greške: 0, STATUS

Izbor uređaja na koji će karakter biti poslat zavisi od sadržaja sistemske promenljive DFLTO $9A (154). Svaki uređaj koji je priključen na serijsku (IEC) vezu i kome je CHKOUT rutinom određeno da radi kao prijemnik (engl. listener) primaće poslate karaktere. Na taj način je moguće istovremeno slanje karaktera na više uređaja.

Primer: Štampanje četrdeset karaktera „*" na štampaču.

```
5 REM *** CHROUT PRIMER ***
10 SYS (8*4096)
20 .OPT 00
30 *= $7000
40 SETNAM = $FFBD
50 SETLFS = $FFBA
60 OPEN = $FFC0
70 CHKOUT = $FFC9
80 CHROUT = $FFD2
90 CLOSE = $FFC3
100 ;
110 :        LDA #0;NAZIV NIJE POTREBAN
120 :        JSR SETNAM
130 :        LDA #3;LOGICKA ADRESA
140 :        LDX #4;BROJ UREDJAJA
150 :        LDY #7;SEKUNDARNA ADRESA
160 :        JSR SETLFS
170 :        JSR OPEN
180 :        LDX #3
190 :        JSR CHKOUT
200 :        LDX #40
210 :        LDA #"*"
220 :LAB     JSR CHROUT
230 :        DEX
240 :        BNE LAB
250 :        LDA #3
260 :        JSR CLOSE
270 :        RTS
280 .END
```

CHRIN
Namena: Učitavanje karaktera sa ulaznog kanala.
Apsolutna adresa: preko vektora $324 (804) na $F157 (61783)
Adresa u tabeli skokova: $FFCF (65487)
Registri za komunikaciju: A: ASCII kôd učitanog karaktera
Registri koji se koriste: A, X
Potrebe za stekom: 7
Pripremne rutine: OPEN, CHKIN
Greške: 0, STATUS

Broj ulaznog uređaja (primarna adresa) nalazi se u sistemskoj promenljivoj DFLTN – $99 (153). Ukoliko nijedan drugi uređaj nije proglašen ulaznim, onda se po uključenju računara, funkcija ulaznog

uređaja automatski dodeljuje tastaturi. Ona se pri tome tretira na poseban način. Ova rutina postavlja kursor, uzima karakter iz bafera tastature i odmah ga ispisuje na ekranu. Ovo se obavlja sve do pritiska tastera RETURN. Pritiskanjem ovog tastera postavlja se indikator koji obeležava dužinu poslednje logičke linije ekranske memorije. Svaki sledeći poziv ove rutine, upisivaće u akumulator kôd sledećeg karaktera iz ekranske memorije sve dok se ne naiđe na RETURN kôd. On ne postoji u ekranskoj memoriji, već se posebno generiše, označavajući kraj logičke linije od 80 karaktera.
Može se izvršiti i upis cele linije odjedanput, ali to zavisi od sadržaja sistemske promenljive CRSW — SDO (208). Ako je njena vrednost 0, izvršiće se upis cele linije, dok će bilo koje druga vrednost omogućiti uzimanje samo sledećeg karaktera iz ekranske memorije i pretvaranje njegovog ekranskog koda u ASCII kôd. Položaj karaktera na ekranu je određen adresom početka linije koja se nalazi u PNT — SD1 (209) i položajem u okviru logičke linije (od 0 do 79) koji je dat u PNTR — SD3 (211).
Sistemska promenljiva STATUS će označiti poslednji bajt (EOF) kada se stigne do poslednjeg karaktera u liniji.
Čitanje karaktera iz datoteke na disku ili traci zahteva proveru da li je pročitani bajt poslednji. Takvu informaciju daje bit 6 sistemske promenljive STATUS. Kod čitanja sekvencijalne datoteke sa trake, krajem se smatra bajt 0, pa CHRIN ispituje jedan bajt unapred.
Ne preporučuje se korošćenje ove rutine za učitavanje karaktera preko RS232 veze jer će se ona obavljati u petlji sve dok spoljašnji uređaj ne pošalje jedan bajt. Ukoliko se ništa ne pošalje, rutina će se vrteti u petlji sve do resetovanja računara.

Primer: Otvaranje datoteke na disku čije se ime unosi preko tastature.

```
5 REM **** CHRIN PRIMER ****
10 SYS (8*4096)
20 .OPT DD
30 *= $7000
40 CHRIN = $FFCF
50 CHROUT = $FFD2
60 SETNAM = $FFBD
70 SETLFS = $FFBA
80 OPEN = $FFC0
90 NAME = $7500
92 CLOSE = $FFC3
95 :TEMP = $88
100 :      LDX #0
110 :LAB   LDA MES1,X
120 :      JSR CHROUT
130 :      INX
140 :      CPX #14
150 :      BNE LAB
160 :      LDX #0
170 :GET   JSR CHRIN
180 :      CMP #$0D
190 :      BEQ EXIT
200 :      STA NAME,X
210 :      INX
220 :      CPX #16
230 :      BNE GET
240 :EXIT  TXA
245 :      LDX #<NAME
246 :      LDY #>NAME
250 :      JSR SETNAM
260 :      LDX #0
270 :LBL   LDA MES2,X
280 :      JSR CHROUT
290 :      INX
300 :      CPX #14
310 :      BNE LBL
320 :      JSR CHRIN
330 :      STA TEMP
340 :      LDX #0
```

```
350 :L3    LDA MES3,X
360 :      JSR CHROUT
370 :      INX
380 :      CPX #19
390 :      BNE L3
400 :      JSR CHRIN
410 :      TAY
420 :      LDX #8
430 :      LDA TEMP
440 :      JSR SETLFS
450 :      JSR OPEN
460 :      LDA TEMP
470 :      JSR CLOSE
475 :      RTS
480 :MES1  .BYTE 13:.ASC "IME DATOTEKE?"
490 :MES2  .BYTE 13:.ASC "LOGICKI BROJ?"
500 :MES3  .BYTE 13:.ASC "SEKUNDARNA ADRESA?"
510 .END
```

GETIN

Namena: Učitavanje pojedinačnog karaktera sa ulaznog kanala.
Apsolutna adresa: preko vektora $32A (810) na $F13E (61758)
Adresa u tabeli skokova: $FFE4 (65508)
Registri za komunikaciju: A: ASCII kôd karaktera (0 ako ništa nije primljeno) i to za RS232 i tastaturu
X,Y – samo za RS232
P: Z =1 ako je A=0
Registri koji se koriste: A,X,Y.
Potrebe za stekom: 7
Pripremne rutine: OPEN, CHKIN
Greške: STATUS

Ova rutina učitava pojedinačni karakter sa uređaja čiji je broj (primarna adresa) u sistemskoj promenljivoj DFLTN – $99 (153). Ukoliko je ostvarena RS232 veza (uređaj broj 2), rutina postavlja A=0 (Z=1) i kada nema nikakvog podatka. To znači da je ulazni bafer prazan. Ukoliko se radi o tastaturi, uzima se jedan karakter iz bafera tastature KEYD – $277 (631) koji je tamo stavljen za vreme IRQ servisne rutine. Ukoliko je i ovaj bafer prazan, postavlja se A=0 (Z=1).
U slučaju svih ostalih uređaja, GETIN se ponaša isto kao i CHRIN.

Primer: Deo programa koji omogućuje vraćanje u bejzik ako se pritisne taster B. Pomoću sličnih programa moguće je organizovati i „menije" tj. izvršavati razne opreracije u zavisnosti od toga koji je taster pritisnut.

```
5 REM ***** GETIN PRIMER *****
10 SYS (8*4096)
20 .OPT OO
30 *= $7000
40 GETIN = $FFE4
50 ;
100 :WAIT JSR GETIN
110 :     CMP #"B"
120 :     BNE WAIT
130 :     JMP ($A002)
140 .END
```

CLRCHN

Namena: Postavljanje standardnih ulaznih (tastatura) i izlaznih (ekran) kanala.
Apsolutna adresa: preko vektora $322 (802) na $F333 (62259)
Adresa u tabeli skokova: $FFCC (65484)
Registri koji se koriste: A,X
Potrebe za stekom: 9

Sistemska promenljiva DFLTN – $99 (153) dobija vrednost 0 (tastatura je ulazni uređaj), a promenljiva DFLTO – $9A (154) vrednost 3 (ekran je izlazni uređaj). Ovim se automatski zatvaraju prethodno otvo-

reni kanali. Ukoliko je ulazni uređaj bio neki od priključenih preko serijske (IEC) veze, izvršava se i UNTLK rutina. Ukoliko je izlazni uređaj bio neki od priključenih preko serijske (IEC) veze, izvršava se i UNLSN rutina.

CLALL

Namena: Zatvaranje svih kanala.
Apsolutna adresa: preko vektora $32C (812) na $F32F (62255)
Adresa u tabeli skokova: $FFE7 (65511)
Registri koji se koriste: A,X
Potrebe za stekom: 11

Zatvaranje svih ulazno/izlaznih kanala se obavlja tako što se sistemskoj promenljivoj LDTND – $98 (152) dodeli vrednost nula. Pošto je sadržajem ove promenljive određen broj otvorenih logičkih datoteka, sa stanovišta operativnog sistema sve datoteke su zatvorene. Međutim, da bi se i fizički zatvorile sve datoteke, potrebno je za svaku izvršiti rutinu CLOSE. Po završetku CLALL automatski se izvršava CLRCHN.

CLOSE

Namena: Zatvaranje datoteke.
Apsolutna adresa: preko vektora $31C (796) na $F291 (620[illegible]
Adresa u tabeli skokova: $FFC3 (65475)
Registri za komunikaciju: A: logički broj datoteke
Registri koji se koriste: A, X, Y
Potrebe za stekom: 2
Greške: 0, 240, STATUS

Stavljanjem logičkog broja datoteke u akumulator i pozivom ove rutine, zatvara se dotična datoteka. Zatvaranje na uređaju povezanom preko RS232 veze osloboдиće deo memorije od 512 bajta (nalazi se na vrhu memorije) koji su korišćeni kao predajni i prijemni baferi. Kada se na kasetofonu izvrši zatvaranje datoteke koja je bila predviđena za upisivanje, automatski se upisuje poslednji blok bez obzira na to što može biti kraći od punih 192 bajta. Ukoliko je pri otvaranju sekundarna adresa bila 2 ili 3, upisuje se još jedan blok sa oznakom EOT (kraj trake).
Pozivom rutine CLOSE, logički broj datoteke se briše iz tabele otvorenih datoteka, a sadržaj sistemske promenljive LDTND – $98 (152) se smanjuje za jedan.

Primer: Videti primer za CHRIN.

LOAD

Namena: Upisivanje u memoriju ili provera zapisa na perifernom uređaju.
Apsolutna adresa: preko vektora $330 (816) na $F49E (62622)
Adresa u tabeli skokova: $FFD5 (65493)
Registri za komunikaciju: A: 0=LOAD; 1=VERIF, X: niži bajt adrese, Y: viši bajt adrese
Registri koji se koriste: A, X, Y
Pripremne rutine: SETLFS, SETNAM
Greške: 0, 4, 5, 8, 9, STATUS

Ova rutina upisuje podatke sa ulaznog uređaja direktno u memoriju. Može se koristiti i za proveru ispravnosti zapisa podataka na perifernom uređaju (disk ili traka) upoređivanjem sa sadržajem u memoriji (engl. VERIFY). Pri tome sadržaj memorije ostaje nepromenjen. Ukoliko je sadržaj akumulatora 0 pre poziva ove rutine, obaviće se LOAD, a ukoliko je 1 obaviće se VERIFY. Pošto LOAD poziva rutinu OPEN, prethodno se moraju izvršiti rutine SETLFS i SETNAM.
Ako je pri pozivu LOAD, sekundarna adresa 0, onda X i Y registri moraju da sadrže početnu adresu od koje se program smešta u memoriju. Ako je sekundarna adresa 1, sadržaj registara X i Y nije bitan, već se program smešta od one adrese koja je zapisana u zaglavlju programa (izvorna adresa). U oba slučaja, po povratku iz ove rutine, X i Y registri će sadržati adresu poslednjeg upisanog bajta.
Napomena: Ne može se obaviti LOAD sa tastature (0), RS232 (2) i ekrana (3).

Primer: Upisivanje u memoriju programa čije se ime unosi preko tastature.

```
5 REM ***** LOAD PRIMER *****
10 SYS (8*4096)
20 .OPT OO
30 *= $7000
40 LOAD = $FFD5
```

```
50 CHROUT = $FFD2
60 CHRIN = $FFCF
70 SETNAM = $FFBD
80 SETLFS = $FFBA
85 NAME = $7500
90 ;
100 :      LDX #0
110 :LAB   LDA MES,X
120 :      JSR CHROUT
130 :      INX
140 :      CPX #15
150 :      BNE LAB
160 :      LDX #0
170 :GET   JSR CHRIN
180 :      CMP #13
190 :      BEQ EXIT
200 :      STA NAME,X
210 :      INX
220 :      CPX #16
230 :      BNE GET
240 :EXIT  TXA
250 :      LDX #<NAME
260 :      LDY #>NAME
270 :      JSR SETNAM
280 :      LDA #0
290 :      LDX #8
300 :      LDY #1
310 :      JSR SETLFS
320 :      JSR LOAD
330 :      RTS
335 :MES .BYTE 13:.ASC "IME PROGRAMA?"
340 .END
```

SAVE

Namena: Snimanje dela memorije na spoljašnju memoriju.
Apsolutna adresa: preko vektora $332 (818) na $F5DD (62941)
Adresa u tabeli skokova: $FFD8 (65496)
Registri za komunikaciju: A: adresa pokazivača početne adrese (pokazivač je na nultoj strani), X: niži bajt krajnje adrese, Y: viši bajt krajnje adrese
Registri koji se koriste: A, X, Y
Potrebe za stekom: SETLFS, SETNAM
Greške: 5, 8, 9, STATUS

Deo memorije, čiji je početak određen sadržajem pokazivača (sam pokazivač je na nultoj strani i njegova adresa je u akumulatoru), a kraj sadržajem X i Y registara, snima se na spoljašnju memoriju (kasetu ili disk).

Pošto ova rutina poziva OPEN, potrebno je prethodno pozvati SETLFS i SETNAM (nije obavezno za kasetofon).

Primer: Program koji snima samog sebe.

```
5 REM ***** SAVE PRIMER *****
10 SYS (8*4096)
20 .OPT OO
30 *= $7000
40 SETNAM = $FFBD
50 SETLFS = $FFBA
60 TXTTAB = 43
70 VARTAB = 45
80 ::SAVE = $FFD8
90 :CLOSE = $FFC3
95 ;
100 :         LDA #11
110 :         LDX #<NAME
120 :         LDY #>NAME
```

```
130 :        JSR SETNAM
140 :        LDA #0
150 :        LDX #8
160 :        LDY #SFF
170 :        JSR SETLFS
180 :        LDA #TXTTAB
190 :        LDX VARTAB
200 :        LDY VARTAB+1
210 :        JSR SAVE
220 :        LDA #0
230 :        JSR CLOSE
240 :        RTS
250 :NAME  .ASC "SAVE PRIMER"
,260 .END
```

Rutine za komunikaciju serijskom vezom

Sve rutine za komunikaciju serijskom (IEC) vezom su deo neke od prethodno opisanih rutina za komunikaciju sa perifernim jedinicama. Međutim, moguće je i njihovo nezavisno pozivanje čime se izbegavaju procedure operativnog sistema za otvaranje i zatvaranje kanala.
Pri ovome, naravno, gubi se informacija o eventualnim greškama. Sistemska promenljiva STATUS je jedina informacija o stanju komunikacije.
Pri upotrebi ovih rutina potrebno je posebnu pažnju obratiti na stanje pojedinih bita u sekundarnoj adresi, jer je ono važno za pravilan rad.
Za svu potrebnu terminologiju i protokol u vezi sa IEEE-488 i IEC vezom, pogledati poglavlje 11. Hardver.

LISTEN
Namena: Naredba adresiranom uređaju da radi kao prijemnik.
Apsolutna adresa: SED0C (60684)
Adresa u tabeli skokova: SFFB1 (65457)
Registri za komunikaciju: A: broj (primarna adresa) uređaja
Registri koji se koriste: A
Greške: STATUS

Ova rutina obavlja operaciju OR između sadržaja akumulatora (broj uređaja) i konstante S20 (32) koja je kôd naredbe LISTEN. Na taj način dobijeni bajt se šalje preko serijske veze. Nakon ovoga, adresirani uređaj je spreman za prijem podataka, pri čemu prvi treba da bude sekundarna adresa.

SECOND
Namena: Slanje sekundarne adrese nakon LISTEN
Apsolutna adresa: SEDB9 (60857)
Adresa u tabeli skokova: SFF93 (65427)
Registri za komunikaciju: A: sekundarna adresa
Registri koji se koriste: A
Potrebe za stekom: 8
Pripremne rutine: LISTEN
Greške: STATUS

Sekundarna adresa se šalje da bi se adresiranom uređaju bliže objasnilo šta da radi. Na primer da li da printer radi sa malim ili velikim slovima, da li da se promeni pero kod pisača ili da li da se otvori komandni kanal kod diska itd.
Sekundarna adresa, isto kao i broj uređaja, šalje se pri $\overline{ATN}=0$ da bi se razlikovala od običnog podatka poslatog preko serijske veze. Vrednost može bit od 0 do 15 i pre poziva se stavlja u akumulator. Biti 5 i 6 moraju da budu 1 dok kod otvaranja datoteke na disku bit 4 i 7 takođe moraju da budu 1.

CIOUT
Namena: Slanje bajta preko serijske veze.
Apsolutna adresa: SEDDD (60893)
Adresa u tabeli skokova: SFFA81 (65448)
Registri za komunikaciju: A: bajt koji se šalje
Potrebe za stekom: 5
Pripremne rutine: LISTEN, SECOND
Greške: STATUS

Ova rutina šalje bajt koji se nalazi u akumulatoru svim uredajima koji su adresirani sa LISTEN. Prethodno se bajt stavlja u bafer i tu ostaje do slanja sledećeg bajta ili do poziva rutine UNLSN.

UNLSN
Namena: Naredba uredajima da prestanu da rade kao prijemni.
Apsolutna adresa: SEDFE (60926)
Adresa u tabeli skokova: SFFAE (65454)
Registri koji se koriste: A
Potrebe za stekom: 8
Greške: STATUS

Ova rutina šalje kôd S3F (63) preko serijske veze, tj. komandu UNLISTEN. Tada svi prijemni uredaji prestaju da primaju podatke iz Komodora i uklanjaju se sa veze. Pre slanja komande UNLISTEN šalje se poslednji bajt iz bafera za CIOUT zajedno sa EOI sekvencom, a zatim ide sama komanda. Da bi se datoteka na disku zatvorila i unele odgovarajuće promene u direktorijum, potrebno je prvo pozvati LISTEN, a zatim i SECOND. Pri tome četvrti bit sekundarne adrese mora biti u stanju 0. Na kraju rutina UNLSN zatvara datoteku.

Primer: Ispisivanje na pisaču.

```
5 REM **LISTEN/UNLISTEN**
10 SYS 8*4096
20 .OPT OO
30 *= $7000
40 SA = 3;IZBOR VELICINE KARAKTERA
50 LISTEN = $FFB1
60 SECOND = $FF93
70 CIOUT = $FFA8
80 UNLSN = $FFAE
90 ;
100 :        LDA #6 ;PLOTER
110 :        JSR LISTEN
120 :        LDA #SA
130 :        JSR SECOND
140 :        LDA #3 ;NAJVECA SLOVA
150 :        JSR CIOUT
160 :        LDX #0
170 :LOOP    LDA NAME,X
180 :        JSR CIOUT
190 :        CPX #7
200 :        BNE LOOP
210 :        JSR UNLSN
220 :        RTS
230 :NAME .ASC "PRIMER":.BYTE $0D
240 .END
```

TALK
Namena: Naredba adresiranom uređaju da radi kao predajnik.
Apsolutna adresa: SED09 (60681)
Adresa u tabeli skokova: SFFB4 (65460)
Registri za komunikaciju: A: sekundarna adresa
Registri koji se koriste: A
Potrebe za stekom: 8
Pripremne rutine: TALK
Greške: STATUS

Ova rutina obavlja operaciju OR izmedu sadržaja akumulatora (broj uređaja) i konstante S40 (64) – kôd za naredbu TALK. Tako dobijeni bajt se šalje preko serijske veze. Nakon ovoga, adresirani uređaj šalje podatak ili čeka sekundarnu adresu nakon čega šalje podatak.
Kada neki uredaj treba sam da pošalje podatak, on to saopštava procesoru preko linije SRQ izazivajući prekid. Procesor tada u prekidnoj rutini šalje naredbu TALK.

TKSA
Namena: Slanje sekundarne adrese nakon TALK.
Apsolutna adresa: SEDC7 (60871)

Adresa u tabeli skokova: $FF96 (65430)
Registri za komunikaciju: A: sekundarna adresa
Registri koji se koriste: A
Potrebe za stekom: 8
Pripremne rutine: TALK
Greške: STATUS

Ova rutina šalje sekundarnu adresu iz akumulatora uređaju adresiranom sa TALK, prethodno postavljajući bite 5 i 6 na jedinicu.

ACPTR
Namena: Čitanje bajta sa adresiranog uređaja preko serijske veze.
Apsolutna adresa: $EE13 (60947)
Adresa u tabeli skokova: $FFA5 (65445)
Registri za komunikaciju: A: primljeni bajt
Registri koji se koriste: A
Potrebe za stekom: 13
Pripremne rutine: TALK, TKSA
Greške: STATUS

Kada adresirani uređaj pošalje bajt preko serijske veze, ova rutina ga čita i stavlja u akumulator. Ako podatak nije poslat, rutina čeka određeno vreme, a zatim javlja grešku.

UNTLK
Namena: Naredba uređajima da prestanu da rade kao predajni.
Apsolutna adresa: $EDEF (60911)
Adresa u tabeli skokova: $FFAB (65451)
Registri koji se koriste: A
Potrebe za stekom: 8
Greške: STATUS

Ova rutina šalje kôd $5F (95) preko serijske veze, tj. komandu UNTALK. Ovu komandu prihvata trenutni predajnik i odmah prestaje sa slanjem podataka računaru.

Primer: Čitanje kanala greške sa disk jedinice.

```
10 SYS 8*4096
20 .OPT OO
30 TALK = $FFB4
40 READST = $FFB7
50 TKSA = $FF96
60 ACPTR = $FFA5
70 UNTLK = $FFAB
100 :         LDA #8
110 :         JSR TALK
120 :         JSR READST
130 :         BNE ERROR
140 :         LDA #%01101111  ; SA=15
150 :         JSR TKSA
160 :         LDY #0
170 : INPUT   JSR ACPTR
180 :         STA $12,Y       ; SMESTANJE PODATAKA
190 :         INY               U BAFER BUF($200)
200 :         CMP #13
210 :         BNE INPUT
220 :         JSR UNTLK
230 : ERROR   RTS
240 .END
```

Rutine za obradu prekida

SETTIM
Namena: Postavljanje časovnika realnog vremena.
Apsolutna adresa: $F6E4 (63204)
Adresa u tabeli skokova: $FFDB (65499)

Registri za komunikaciju: A: najviši bajt, X: niži bajt, Y: najniži bajt
Potrebe za stekom: 2

Sistemska promenljiva TIME – SA0 (160) nalazi se na nultoj strani i sastoji se od tri bajta. Oni čine brojač čiji se sadržaj povećava za jedan prilikom svakog standardnog prekida programa (IRQ svakih 1/60 sekunde). Ovom rutinom se sadržaj brojača postavlja na početnu vrednost.

RDTIM
Namena: Čitanje časovnika realnog vremena.
Apsolutna adresa: SF6DD (63197)
Adresa u tabeli skokova: SFFDF (65505)
Registri za komunikaciju: A: najviši bajt, X: niži bajt, Y: najniži bajt
Registri koji se koriste: A, X, Y
Potrebe za stekom: 2

Ova rutina čita sadržaj sistemske promenljive TIME – SA0 (160) i smešta odgovarajuće bajtove u registre mikroprocesora.

Primer: Potprogram za generisanje kašnjenja od 1 sekunde.

```
5 REM **SETTIM PRIMER**
10 SYS 8*4096
20 .OPT 00
30 *= $7000
40 SETTIM = $FFDB
50 RDTIM = $FFDE
60 ;
100 :DELAY LDA #0
110 :       TAX
120 :       TAY
130 :       JSR SETTIM
140 :WAIT   JSR RDTIM
150 :       CMP #$3C
160 :       BCC WAIT
170 :       RTS
180 .END
```

UDTIM
Namena: Ažuriranje časovnika realnog vremena.
Apsolutna adresa: SF69B (63131)
Adresa u tabeli skokova: SFFEA (65517)
Registri koji se koriste: A, X
Potrebe za stekom: 2
Ova rutina je deo programa za obradu prekida koji se poziva svakih 1/60 sekunde. Ona povećava vrednost sistemske promenljive TIME – SA0 (160) za 1. Ukoliko je dostignuta granica od 5184000 (24 časa), TIME se postavlja na nulu. UDTIM, osim svoje standardne funkcije, očitava stanje STOP tastera. Ukoliko je on u tom trenutku pritisnut, vrednost sistemske promenljive STKEY – S91 (145) se postavlja na vrednost S7F (127).

STOP
Namena: Testiranje STOP tastera.
Apsolutna adresa: preko vektora S328 (808) na SF6ED (63213)
Adresa u tabeli skokova: SFFE1 (65505)
Registri za komunikaciju: P: Z=1 ako je STOP pritisnut
Registri koji se koriste: A, X

Ova rutina ispituje sadržaj sistemske promenljive STKEY – S91 (145) i ukoliko je njen sadržaj S79 (127) postavlja indikator Z na jedinicu. To znači da je STOP taster bio pritisnut pri poslednjem pozivu rutine UDTIM. Posle ovoga poziva se rutina CLRCHN koja postavlja tastaturu kao ulazni, a ekran kao izlazni uređaj i na kraju se prazni bafer tastature.

Primer: Vraćanje u bejzik ukoliko je pritisnut STOP taster.

```
5 REM ** STOP PRIMER **
10 SYS 8*4096
20 .OPT OO
30 *= $7000
40 STOP = $FFE1
50 ;
60 :TEST   JSR STOP
70 :       BNE TEST
80 :       JMP ($A002)
90 .END
```

SCNKEY

Namena: Očitavanje tastature.
Apsolutna adresa: $EA87 (60039)
Adresa u tabeli skokova: $FF9F (65439)
Registri koji se koriste: A, X, Y
Potrebe za stekom: 5
Pripremne rutine: IOINIT (ukoliko nije izvršena po uključenju računara)

Ovo je rutina koju poziva program za obradu prekida. Prekid nastupa svake 1/60 sekunde, a generiše ga tajmer A u CIA 1 (za detalje o ovom, kao i čitanju tastature videti poglavlje 11. Hardver).
Preko CIA 1 čita se kôd (redni broj) tastera i stavlja se u SFDX — $CB (203) pa se u zavisnosti od toga da li je pritisnut i neki od kontrolnih tastera, ovaj broj koristi za očitavanje odgovarajućeg PETASCII koda iz jedne od sledećih tabela:

1. standardna tabela — $EB81 (60289)
2. taster+SHIFT — $EBC2 (60354)
3. taster+C= — $EC03 (60419)
4. taster+CTRL — $EC78 (60536)

Na primer, ako je pritisnut taster RETURN, dobija se kôd 1 pa se iz standardne tabele (nije pritisnut ni jedan kontrolni taster) doblja PETASCII kôd $0D (13). Tabelu bira rutina u koju se ulazi preko vektora KEYLOG — $28F (655) tako da korisnik može i sam da definiše svoje tabele tj. svoje kodove za pritisnute tastere. Pri tome treba napisati i novu rutinu za izbor tabele i njenu adresu staviti u KEYLOG.
Dobijeni PETASCII kôd se stavlja u bafer tastature KEYD — $277 (631). Bafer je organizovan kao red (engl. queue) tj. čine ga spojene lokacije $277 — $280 (631 — 640) organizovane kao FIFO (engl. first in first out). U njega može da se smesti maksimalno deset karaktera, a može ih biti i manje ukoliko je to određeno sadržajem sistemske promenljive XMAX — $289 (649). Broj karaktera koji čekaju na obradu određuje sistemska promenljiva NDX — $C6 (198). Ukoliko karakter treba prikazati, on se pomera na kraj bafera — $277 (631). Prvi karakter stavljen u bafer biće pročitan pozivom GETIN rutine.
Promenom sadržaja sistemskih promenljivih XMAX i NDX mogu se postići interesantni efekti kao što je dinamička promena teksta bejzik programa (u toku izvršavanja programa) kao i povezivanje (engl. merge) sekvencijalnih datoteka itd.

Primer: Ukoliko je pritisnut taster „B" vratiti se u bejzik.

```
5 REM ** SCNKEY PRIMER **
10 SYS 8*4096
20 .OPT OO
30 *= $7000
40 SCNKEY = $FF9F
50 GETIN = $FFE4
60 ;
100 :       SEI
110 :LOOP   JSR SCNKEY
120 :       JSR GETIN
130 :       CMP #"B"
140 :       BNE LOOP
150 :       CLI
160 :       JMP ($A002)
170 .END
```

PLOT
Namena: Čitanje ili postavljanje pozicije kursora.
Apsolutna adresa: $E50A (58634)
Adresa u tabeli skokova: $FFF0 (65520)
Registri za komunikaciju: X: broj reda, Y: broj kolone, P: C=1 čitanje: C=0 postavljanje
Registri koji se koriste: A, X, Y
Potrebe za stekom: 2

Ukoliko se pre poziva ove rutine postavi C=1 dobiće se u X registru broj reda (0 – 24), a u Y registru broj kolone (0 – 39) trenutne pozicije kursora. Ako treba postaviti kursor na određeno mesto, prethodno treba postaviti C=0, a u X i Y registre upisati broj reda i broj kolone. Trenutna pozicija kursora inače se nalazi u sistemskim promenljivama TBLX – $D6 (214) – broj linije i PNTR – $D3 (211) – broj kolone.

Primer: Potprogram LNTST koji skraćuje liniju fizičkog ekrana na 20 karaktera.

```
5 REM *** PLOT PRIMER ***
10 SYS 8*4096
20 .OPT OO
30 *= $7000
40 PLOT = $FFF0
50 SCROLL = $EBEA
100 :        SEC
110 :        JSR PLOT
120 :        CPY #20;DA LI JE KURSOR U DVADESETOJ KOLONI
130 :        BEQ NXT
140 :        RTS
150 :NXT     INX
156 :        CPX #12;DA LI JE KURSOR U DVANAESTOM REDU
157 :        BNE OK
159 :        JSR SCROLL;AKO DA,POZOVI SCROLL RUTINU
160 :OK      LDY #0
170 :        CLC
180 :        JSR PLOT
190 :        RTS
200 .END
250 REM PISANJE U GORNJOJ LEVOJ CETVRTINI EKRANA
251 REM MAX BROJ LINIJA 12
252 REM MAX BROJ KOLONA 20
300 PRINT"{CLR}"
310 GET A$
320 SYS 7*4096
330 PRINT A$;
340 GOTO 310
```

8.4.2 Organizacija operativnog sistema

Kernal je smešten u ROM koji zauzima adresni prostor od $E000 (57344) do $FFFF (65535). Međutim deo ovog ROM-a zauzimaju neke rutine bejzik interpretera, tako da Kernal praktično započinje od adrese $E4DA (58586). U opisu koji sledi, rutine su date po rastućim adresama. Rutine koje nisu dokumentovane tj. u koje ne može da se uđe preko tabele skokova, dodatno su opisane. Ove rutine mogu se nezavisno koristiti jer ih je najveći deo u obliku potprograma. Upotrebom nekog monitorskog programa (npr. Monitor 64) može se pregledati svaka rutina i uneti eventualne izmene. Naravno, pre unošenja izmena potrebno je kopirati sadržaj Kernal ROM-a u RAM koji se nalazi na paralelnim lokacijama, a zatim isključiti ROM iz adresnog prostora i uključiti RAM. To se postiže postavljanjem bita 1 (signal $\overline{HIRAM}$) u ulazno/izlaznom registru mikroprosecora (adresa $1) na nulu.

Ukoliko će se u daljem radu koristiti bejzik potrebno je izvršiti i kopiranje bejzik ROM-a u RAM jer se pomoću $\overline{HIRAM}$ signala i on isključuje.

```
10 REM PRENOS BEJZIK INTERPRETERA U RAM
20 FOR I=40960 TO 49151
30 POKE I,PEEK(I)
40 NEXT
50 REM PRENOS KERNALA U RAM
60 FOR I=57344 TO 65535
70 POKE I,PEEK(I)
80 NEXT
85 REM HIRAM = 0
90 POKE 1,PEEK(1) AND 253
```

$E4DA (58586)
Postavljanje kolor RAM-a na vrednosti iz registra pozadine broj 1 (engl. background color register 1). Ova rutina je dodatak kasnijim verzijama Kernala, a deo je rutine za brisanje jedne linije ekranske memorije.

$E4E0 (58592)
Pauza od 8.5 sekundi posle nalaženja programa ili datoteke na traci. Ukoliko se u međuvremenu pritisne neki taster, prekida se ova rutina i počinje punjenje programa (podataka) u memoriju.

$E4EC (58604)
Tabela brzine slanja bita (engl. baud rate) za PAL sistem. Zbog razlike u frekvenciji sistemskog takta za PAL i NTSC sistem, potrebno je odrediti broj kojim će se podeliti frekvencija takta da bi se dobila odgovarajuća brzina prenosa bita. Ovo je tabela svih tih brojeva za sve brzine prenosa bita koje može da koristi Komodor u PAL sistemu.

Rutine ekranskog editora

$E500 (58624) — IOBASE

$E505 (58629) — SCREEN

$E50A (58634) — PLOT

$E518 (58648)
Ovo je CINT rutina u starijim verzijama Kernala koja se ne bavi ispitivanjem raster registra VIC-a niti postavljanjem tajmera u CIA 1. Ona je deo nove CINT rutine.

$E544 (58692)
Inicijalizacija link tabele LDTB1 — ($D9 (217), brisanje ekrana i postavljanje kolor RAM-a na boju pozadine.

$E566 (58726)
Postavljanje sistemskih promenljivih PNTR — $D3 (211) i TBLX — $D6 (214) na vrednost 0 tj. ekranskog pokazivača u gornji levi ugao.

$E56C (58732)
Postavljanje pokazivača PNT — $D1 (209) na adresu prvog bajta trenutne logičke linije. Ova rutina se pri tome koristi sadržajem link tabele, tj. proverava da li su dve fizičke linije spojene u jednu logičku.

$E5A0 (58784)
Postavljanje DFLTO — $9A (154) na 3 i DFLTN — $99 (153) na 0.

$E5A8 (58792) — INITV
Kopiranje vrednosti iz tabele $ECB9 (60601) u registre VIC-a.

$E5B4 (58804) — LP2
Ova rutina uzima jedan karakter iz bafera tastature I stavlja ga u akumulator, pomera sve preostale karaktere za po jedno mesto u redu i umanjuje za jedan sadržaj sistemske promenljive NDX — $C6 (198) koja pokazuje koliko karaktera čeka na obradu.

$E5CA (58826)
Ovo je deo rutine CHRIN koji uključuje kursor na ekranu, uzima karaktere sa tastature i prikazuje ih na ekranu sve do pritiska tastera RETURN. Takođe ispituje da li su istovremeno pritisnuti SHIFT i RUN/STOP. Ukoliko jesu, string sa adrese $ECE7 (60647) prebacuje se u bafer tastature.

$E632 ($58930)
Deo rutine CHRIN koji čita karaktere sa tastature ili sa ekrana u zavisnosti od stanja sistemske promenljive CRSW — $D0 (208).

$E684 ($59012)
Rutina koja ispituje da li su otvoreni znaci navoda. Ako jesu, sistemska promenljiva QTSW — $D4 (212) se postavlja na vrednost različitu od nule.

$E691 ($59025)
Rutina koja u ekransku memoriju stavlja one karaktere koji mogu da se prikažu na ekranu.

$E6B6 ($59062)
Pomeranje kursora na ekranu za jedno mesto udesno. Ukoliko je u pitanju poslednje mesto u poslednjem redu ili poslednja linija sa pritisnutim RETURN tasterom, obavlja se pomeranje sadržaja ekrana za jedan red nagore (engl. scroll).

$E701 ($59137)
Vraćanje kursora u novi red posle četrdesete pozicije u prethodnom redu.

$E716 ($59158) — SCPNT
Ovo je rutina koju poziva CHROUT kada treba da ispiše znak na ekranu. Međutim ona se može pozivati i kao samostalna, pri čemu se izbegava standardna procedura Kernala za slanje karaktera na bilo koji uređaj.
Ova rutina testira da li se karakter, čiji se ASCII kôd u akumulatoru, može ispisati na ekranu. Ukoliko može, ispisuje ga, a ukoliko ne može, obavlja operaciju koju taj karakter definiše (pomeranje kursora, promena boje, brisanje ekrana i slično).

$E8CB ($59595)
Ova rutina koristi CHROUT za ispitivanje da li kôd karaktera koji treba da se ispiše predstavlja naredbu za promenu boje ispisa (npr. CTRL—1).

$E8D1 ($59601)
Tabela PETASCII kodova za boje.

$E8EA ($9626) — SCROLL
Rutina za pomeranje sadržaja ekrana nagore za jedan red (engl. scroll). Ukoliko je poslednja linija ispisana na ekranu, pre ispisivanja sledeće izvršava se ova rutina. Ukoliko se logička linija na vrhu sastoji od dve fizičke linije, sadržaj ekrana se pomera za dve linije. Držanje CTRL tastera generiše izvesno kašnjenje po izvršenju rutine.

$E965 ($59848)
Ubacivanje prazne linije na ekranu.

$E9C8 ($59848)
Pomeranje jedne linije ekranske i kolor memorije na gore.

$E9E0 ($59872)
Postavljanje pokazivača (pointera) EAL — $AE (174) na adresu kolor RAM-a koja odgovara privremenoj adresi linije u SAL — $AC (172).

$E9FO ($59888)
Stavljanje adrese prvog bajta ekranske linije određene ofsetom u X registru u PNT — $D1 (209).

$E9FF ($59903)
Ova rutina upisuje prazna mesta u celoj ekranskoj liniji, a odgovarajuću liniju u kolor RAM-u postavlja na boju pozadine.

$EA13 ($9923)
Postavljanje brojača vremena treptaja kursora (lokacija BLNCT — $CD (205), a zatim stavljanje karaktera iz akumulatora u lokaciju na koju pokazuje PNT — $D1 (209) i boje iz X registra u lokaciju na koju pokazuje USER — $F3 (243).

$EA24 ($59940)
Sinhronizacija pokazivača USER — $F3 (243) i PNT — $D1 (209) na korespondentne lokacije u ekranskoj i kolor memoriji.

Rutine za obradu maskirajućeg prekida (IRQ)

Po nastanku maskirajućeg prekida, vrši se indirektan skok preko hardverskog vektora SFFFE na sledeću rutinu:

```
        PHA; čuvanje registara A, X, Y na steku
        TXA
        PHA
        TYA
        PHA
        TSX
        LDA S104, X; direktno uzimanje P registra bez promene sadržaja steka
        AND S10; da li je B bit u P registru=0?
        BEQ IRQ; ako da, onda je IRQ
        JMP (CBINV); ako ne, onda je BRQ
IRQ     JMP (CINV)
```

Vektor CINV — $314 (788) posle inicijalizacije sistema sadrži adresu SEA31 (S99S3) gde počinje rutina za obradu maskirajućeg prekida.

SEA31 (S9953) — IRQ
Glavna rutina za obradu IRQ prekida. Ona inkrementira sadržaj softverskog časovnika TIME — SA0 (160), podržava treptanje kursora, drži uključen motor kasetofona ukoliko je pritisnut odgovarajući taster i na kraju poziva rutinu za očitavanje tastature. Prekid, kojim se ulazi u ovu rutinu, nastaje svakih 1/60 sekunde.

SEA87 (60039) — SCNKEY

SEAE0 (60128)
Deo SCNKEY rutine koji dekoduje pritisnut taster i njegov odgovarajući PETASCII kôd stavlja u bafer tastature. Ukoliko je taster isti kao prethodno pritisnuti, ova rutina ispituje da li ga treba ponavljati bez otpuštanja.

SEB48 (60232)
Rutina koja bira odgovarajuću tabelu za dekodovanje zavisno od toga da li je pritisnut neki od kontrolnih tastera: SHIFT, CTRL, C= ili nijedan.

SEB79 (60281)
Tabela vektora dekoderskih tabela.

SEB81 (60289)
Tabela za dekodovanje pritisnutog tastera.

SEBC2 (603S4)
Tabela za dekodovanje pritisnutog tastera uz pritisnut SHIFT.

SEC03 (60419)
Tabela za dekodovanje pritisnutog tastera uz pritisnut C=.

SEC44 (60484)
Deo CHROUT rutine koja određuje da li treba odabrati skup karaktera za mala/velika slova ili velika slova/grafika.

SEC5E (60S10)
Postavljanje indikatora za omogućenje ili onemogućenje prebacivanja sa jednog na drugi skup karaktera pritiskom na C= i SHIFT.

SEC78 (60S36)
Tabela za dekodovanje pritisnutog tastera uz pritisnut CTRL.

Ulazno izlazne rutine

SECB9 (60601)
Tabela vrednosti koje se upisuju u registre VIC-a pri inicijalizaciji.

SECE7 (60647)
Tekst koji se stavlja u bafer tastature kada se istovremeno pritisnu tasteri SHIFT i RUN. Standardno je to LOAD CR RUN gde je CR — CHR$(13).

$ECF0 (60656)
Tabela nižih bajtova adresa ekranskih linija. Viši bajtovi se dobijaju iz kombinacije vrednosti ekranske link tabele LDTB1 – D9 (217) i pokazivača (pointera) ekranske memorije HIBASE – $288 (648).

$ED09 (60681) – TALK

$ED0C (60684) – LISTEN

$ED11 (60689)
Slanje komandnog koda preko serijske veze.

$ED40 (60736)
Slanje bajta iz bafera BSOUR – $95 (149) preko serijske veze.

$EDB0 (60848)
Ova rutina generiše kôd greške DEVICE NOT PRESENT ukoliko uređaj nije priključen.

$EDB9 (60857) – SECOND

$EDC7 (60871) – TKSA

$EDDD (60893) – CIOUT

$EDEF (60911) – UNTLK

$EDFE (60926) – UNLSN

$EE13 (60947) – ACPTR

$EE85 (61061)
Postavljanje linije CLOCK serijske veze na nulu.

$EEBE (61070)
Postavljanje linije CLOCK serijske veze na jedinicu.

$EE97 (61079)
Postavljanje linije DATA serijske veze na nulu.

$EEA0 (61088)
Postavljanje linije DATA serijske veze na jedinicu.

$EEA9 (61097)
Učitavanje bita sa serijske veze. Bit učitan sa DATA linije se stavlja u C, a sa CLOCK linije u indikator N u P registru.

$EEB3 (61107)
Kašnjenje od jedne milisekunde.

$EEBB (61115)
Ova rutina poziva rutinu za obradu NMI prekida kada treba da pošalje bit preko RS232 veze.

$EF2E (61230)
Ova rutina postavlja na određenu vrednost bite u status registru RSSTAT – $297 (663) ukoliko nastane greška u komunikaciji RS232 veze.

$EF4A (61258)
Rutina koja uzima broj bita podataka iz kontrolnog registra i stavlja ga u X registar za upotrebu od strane drugih RS232 rutina.

$EF59 (61273)
Ovu rutinu poziva rutina za obradu NMI prekida kada treba da primi bit preko RS232 veze.

$EF97 (61335)
Stavljanje primljenog bajta u RS232 bafer. Zatim se ispituje da li je došlo do greške u parnosti, sinhronizaciji rama ili do prekida. Na kraju se vrše pripreme za prijem sledećeg bajta.

$EFE1 (61409)
Deo CHROUT rutine za RS232 vezu.

$F04D (61517)
Deo CHKIN rutine za RS232 vezu.

SF086 (61574)
Deo GETIN rutine za RS232 vezu.

SF0A4 (61604)
Ukoliko se obavlja ulazno/izlazna operacija preko veze za kasetofon ili serijske veze gde postoji vremenski kritičan protokol, ova rutina onemogućuje generisanje NMI prekida od strane CIA 2 tj. onemogućuje RS232 vezu.

SF0BD (61629)
Tabela ASCII tekstova kontrolnih poruka Kernala. Poslednji bajt svake poruke ima postavljen sedmi bit na jedinicu . (ASCII + S80)
Poruke su:
I/0 ERROR
SEARCHING
FOR
PRESS PLAY ON TAPE
PRESS RECORD & PLAY ON TAPE
LOADING
SAVING
VERIFYING
FOUND
OK

SF12B (61739)
Ova rutina proverava da li je dozvoljeno prikazivanje Kernal-ovih poruka o greškama ispitujući sadržaj lokacije MSGFLG — S9D (157). Ukoliko jeste, prikazuje poruku indeksiranu Y registrom.

SF13E (61758) — GETIN
SF157 (61783) — CHRIN
SF1CA (61898) — CHROUT
SF20E (61966) — CHKIN
SF250 (62032) — CHKOUT
SF291 (62096) — CLOSE
SF30F (62223)
Ova rutina traži položaj logičke datoteke u tabeli LAT — 259 (601).

SF31F (62239)
Postavljanje vrednosti u sistemske promenljive LA — SB8 (184) — broj logičke datoteke, SA — S89 (158) — sekundarna adresa, FA — SBA (186) — primarna adresa (broj uređaja).

SF32F (62255) — CLALL
SF333 (62259) — CLRCHN
SF34A (62282) — OPEN
SF49E (62622) — LOAD
SF5DD (62941) — SAVE
SF69B (63131) — UDTIM
SF6DD (63197) — RDTIM
SF6E4 (63204) — SETTIM
SF6ED (63213) — STOP

SF6FB (63227)
Ova rutina obrađuje greške koje nastaju kod ulazno izlaznih operacija. Ona prvo poziva CLRCHN, a zatim, ako je bit 6 na lokaciji MSGFLG — S9D (157) na jedinici, ispisuje poruku I/0 ERROR sa propratnim brojem greške. Posle ovoga, postavlja se C indikator u P registru na jedinicu, a broj greške se stavlja u akumulator.

SF72C (63276)
Upis bloka zaglavlja sa trake, provera tipa datoteke i ispisivanje poruke FOUND sa imenom datoteke.

SF76A (63338)
Zapisivanje broja zaglavlja na traku.

SF7D0 (63440)
Punjenje X i Y registra adresom bafera kasetofona.

$F707 (63447)
Postavljanje pokazivača (pointera) ulazno/izlaznog prostora na početnu i krajnju adresu bafera kasetofona.

$F7EA (63466)
Traženje imena datoteke na traci.

$F817 (63511)
Kontrola stanja tastera na kasetofonu. Ukoliko nijedan nije pritisnut ispisuje se poruka PRESS PLAY ON TAPE, a ako jeste ispisuje se OK.
Pošto se u ovu rutinu ulazi posle provere da li se radi o direktnom načinu rada, ispisivanje poruka se ne može sprečiti menjanjem sadržaja MSGFLG — $9D (157).

$F82E (63534)
Ispitivanje da li je pritisnut taster na kasetofonu.

$F838 (63544)
Rutine za vremensku sinhronizaciju čitanja i upisivanja na kasetofon kao i kontrolu motora.

$F8D0 (63696)
Ispituje da li je pritisnut taster STOP za vreme ulazno/izlazne operacije. Ukoliko jeste, prekida operaciju.

$F8E2 (63714)
Postavljanje vrednosti tajmera A iz CIA 1 na vrednosti za sinhronizaciju pri radu sa kasetofonom.

$F92C (63788)
Rutine koje su deo rutine za obradu maskirajućeg prekida (IRQ). Korlste se za čitanje podataka sa trake.

$FB8E (64398)
Prebacivanje adrese za LOAD/SAVE u pokazivač SAL — $AC (172)

$FBA6 (64422)
Ova rutina postavlja tajmer B u CIA 1 i uključuje izlaznu liniju za kasetofon (lokacija $1C1, bit 3).

$FBC8 (64456)
Deo rutine za obradu maskirajućeg prekida (IRQ) za zapisivanje podataka na traku.

$FC93 (64659)
Na kraju ulazno-izlazne operacije sa kasetofonom, ova rutina vraća sadržaj ekrana i zaustavlja motor. Zatim resetuje tajmer A iz CIA 1 tj. postavlja vrednost za generisanje prekida svakih 1/60 sekunde, vraća stari IRQ vektor koji pokazuje na rutine za časovnik i očitavanje tastature.

$FCBB (64696)
Završetak ulazno/izlazne operacije sa kasetofonom.

$FCCA (64714)
Isključivanje motora kasetofona.

$FCD1 (64721)
Ova rutina upoređuje trenutnu adresu za učitavanje/zapisivanje na traku sa krajnjom adresom.

$FCDB (64731)
Uvećavanje sadržaja pokazivača trenutne adrese za učitavanje/zapisivanje na traku za jedan.

$FCE2 (64738) — RESET
$FD02 (64770) — AOINT
$FD15 (64789) — RESTOR
$FD1A (64794) — VECTOR

$FD30 (64816)
Tabela standardnih vektora za ulazno/izlazne rutine. Ovih 16 vektora se prebacuje u oblast RAM-a $314 — $333 (788 — 819).

$FD50 (63848) – RAMTAS
$FD9B (64923) Tabela IRQ vektora:
SFD9B $FC6A (64618) – CASS WRITE 1
SFD9D $FBCD (64461) – CASS WRITE 2
SFD9F $EA31 (59953) – KBD SCAN
SFDA1: $F92C (63788) – CASS READ
$FDA3 (64931) – IOINIT
$FDF9 (65017) – SETNAM
$FE00 (65024) – SETLFS
$FE07 (65031) – READST
$FE18 (65048) – SETMSG
$FE21 (65057) – SETTMO
$FE25 (65061) – MEMTOP
$FE34 (65076) – MEMBOT
$FE43 (65091) – NMI

U ovu rutinu se direktno ulazi preko hardverskog NMI vektora $FFFA (65530). Ona prvo zabranjuje IRQ prekid postavljanjem I indikatora na jedinicu. Nakon toga vrši se indirektan skok preko RAM vektora $318 (792) u rutinu za obradu NMI. U standardnoj verziji Kernala ona je već na sledećoj memorijskoj lokaciji. Prvo se ispituje da li je NMI prekid nastao od strane RS232 uređaja. Ako nije, usvaja se da je bio pritisnut taster RESTORE. Ispituje se prisustvo autostart ROM-a pa ako postoji, vrši se indirektni skok preko vektora $8002 (32770). Ukoliko ovaj ROM ne postoji, ispituje se da li je pritisnut STOP taster, pa ako jeste ide se na rutinu za obradu BRK.
Ukoliko je uređaj sa RS232 veze izazvao NMI prekid, sve prethodno se preskače pa se ispituje da li podatak treba da se šalje ili da se prima.

$FE66 (65126) – BRK
Ova rutina se izvršava kada su pritisnuti tasteri STOP i RESTORE ili kada se naiđe na naredbu BRK. Ona poziva inicijalizacione rutine RESTOR, IOINIT i deo CINT, a zatim indirektno ulazi u bejzik preko vektora $A002 (40962).

$FE72 (65138)
Ovo je deo NMI rutine za obradu ulaza i izlaza preko RS232 veze.

$FEC2 (65218)
Tabela brzine slanja bita (engl. baud rate) za NTSC sistem.

$FED6 (65238)
Deo NMI rutine za primanje jednog bita preko RS232 veze.

$FF48 (65352)
Zajednička ulazna adresa za IRQ i BRK. Za detaljnije podatke pogledati rutine za obradu IRQ na adresi $EA31.

$FF5B – $FF7F (65371 – 65407)
Izmene i dopune novijih verzija operativnog sistema.

$FF5B (65371) – CINT

$FF80 (65408)
Na ovoj lokaciji se nalazi bajt za identifikaciju operativnog sistema. Prve verzije imaju oznaku $AA (170) dok kasnije verzije imaju oznaku $00 (0) ili $03 (3).

$FF81 – $FFF5 (65409 – 65525) – tabela skokova (engl. Kernal jump table).

$FFFA (65530)
Hardverski NMI vektor. Pokazuje za adresu $FE43 (65091).

$FFFC (65532)
Hardverski RESET vektor. Pokazuje na adresu $FCE2 (64738).

$FFFE (65534)
Hardverski IRQ/BRK vektor. Pokazuje na adresu $FF48 (65352).

9
Zvuk

9.1 ELEMENTI SINTESAJZERA

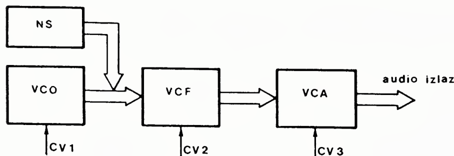
Sintesajzer (engl. synthesizer) je uređaj koji je poslednjih godina ušao u svakodnevnu upotrebu. Najčešće se koristi kao muzički instrument koji imitira druge instrumente ili generiše nove zvuke koji ne postoje u prirodi.

Osnovu svakog sintesajzera čini naponski kontrolisani oscilator — VCO (engl. voltage controlled oscillator). Izlazna frekvencija VCO-a zavisi od veličine kontrolnog napona. Talasni oblici izlaznog napona mogu biti različiti. Najčešće su trougaonog, testerastog ili četvrtastog oblika. Kao izvor signala može se koristiti i generator šuma NS (engl. noise source). Ukoliko postoji više nezavisnih VCO, sintesajzer je višeglasni. Neki sintesajzeri imaju više VCO-a, a ipak su jednoglasni jer su im kontrolni naponi zajednički.

Drugi važan element je naponsko kontrolisani filter — VCF (engl. voltage controlled filter). Postoji više različitih tipova VCF-a, ali se najčešće koristi niskopropusni filter. Njegova granična frekvencija zavisi od kontrolnog napona.

Pomoću filtera se može menjati sadržaj harmonika u talasnom obliku koji generiše VCO.

Treći element je naponsko kontrolisani pojačavač — VCA (engl. voltage controlled amplifier). Njegovo pojačanje zavisi od veličine kontrolnog napona.



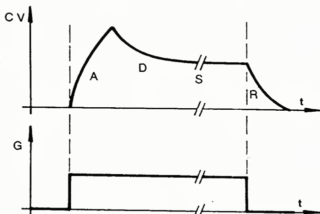
Sl. 9. 1. Standardni elementi sintesajzera

Na slici je prikazana veza između ovih elemenata. Širim linijama je obeležen put audio signala, dok su strelicama obeleženi kontrolni naponi (CV1, CV2 i CV3). Generatori kontrolnog napona mogu biti:

— Generator jednosmernog napona. Pomoću njega se najčešće bira frekvencija VCO (visina tona) ili granična frekvencija filtera.

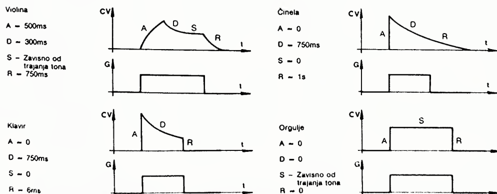
— Generator niskih frekvencija LFO (engl. low frequency oscillator). Ovaj oscilator daje različite talasne oblike koji mogu biti i vrlo niskih frekvencija (ispod 0.1Hz). Ukoliko se sa frekvencijom od oko 5Hz kontroliše VCO, dobija se vibrato tj. frekventna modulacija, a ako se to primeni na VCA, dobija se tremolo tj. amplitudna modulacija. LFO može da kontroliše i VCF — to je poznati WAH-WAH efekat.

— Generator obvojnice (engl. envelope generator). Amplituda svakog zvuka dinamički se menja od trenutka kada nastane do trenutka kada nestane. Skup vršnih vrednosti amplitude signala u ovom intervalu naziva se obvojnica.



Sl. 9. 2. Načini generisanja i pojedine faze obvojnice zvuka

Kontrolni logički signal G (engl. GATE — kapija) startuje generisanje obvojnice svojim prelaskom na jedinicu. Prvo nastaje faza porasta — A (engl. attack), a za njom faza opadanja do srednje vrednosti — D (engl. decay). Faza srednje vrednosti — S (engl. sustain) traje sve dok je kontrolni signal G na jedinici, što može da bude proizvoljno dugo. Kada G postane nula, nastaje faza opadanja — R (engl. release). Generator obvojnice se nikada ne koristi za kontrolu VCO, a ukoliko se primeni za kontrolu VCF dobijaju se poznati sintetički („svemirski") zvuci.



Sl. 9. 3. Tipični oblici obvojnica zvuka nekih muzičkih instrumenata

Naravno, za potpuno imitiranje pojedinih instrumenata potrebno je podesiti visinu tona kao i odabrati talasni oblik (VCO), a takođe i odrediti boju tona (VCF).

9.2 GENERISANJE ZVUKA U KOMODORU

Zahvaljujući postojanju specijalizovanog integrisanog kola, audio kontrolera 6581 — SID (engl. sound interface device), Komodor poseduje izvanrednu hardversku podršku za generisanje zvuka. Nažalost, to se ne može reći i za softver. Komodorov bejzik uopšte ne podržava zvuk, pa se programiranje mora obaviti na mašinskom jeziku. 6581 je programabilni troglasni sintesajzer. Svakom od tri oscilatora priključen je po jedan VCA sa sopstvenim generatorom obvojnice. VCF je zajednički za sva tri glasa, a kao LFO se može koristiti oscilator 3. Postoji i generator šuma kao i izlazni sabirni pojačavač. Naravno, ovde se ne radi o naponskoj već o digitalnoj kontroli pojedinih elemenata. Unutar SID-a nalaze se D/A konvertori koji digitalne reči pretvaraju u analogne kontrolne napone. Ovi naponi mogu da imaju samo diskretne vrednosti.

Registri audio kontrolera nalaze se u adresnom prostoru mikroprocesora počev od lokacije $D400 (54272) i ima ih ukupno 28.

$D400 (54272) — FREQLO
Kontrola frekvencije oscilatora 1, niži bajt.

$D401 (54273) — FREQHI
Kontrola frekvencije oscilatora 1, viši bajt. Ukoliko se radi sa temperovanim sistemom (koncertno A=440Hz), razmak između dva susedna polutona iznosi 2, pa se kontrolne reči za pojedine tonove mogu odrediti na osnovu formule:

FS=INT (FR*17.0284+0.5) gde je:
FR=440*(2 ↑(I−9/12)), a I ima sledeće vrednosti:

I	ton	I	ton
0	C	6	F #
1	C#	7	G
2	D	8	G #
3	D#	9	A
4	E	10	A #
5	F	11	H

Ukoliko je potrebno preći u višu oktavu, FR treba pomnožiti sa 2, a u nižu, podeliti ga sa 2. Sa 65535 različitih frekvencija moguće je pokriti 7 oktava.

$D402 — $D403 (54274 — 54275) — PWLO / PWHI
Ova dva registra čine jedinstveni 12-bitni registar (biti 4 do 7 registra PWHI se ne koriste). Ukoliko je izabrani talasni oblik oscilatora 1 četvrtast, sadržaj ovog registra određuje odnos između trajanja impulsa i pauze (engl. duty dycle). Širina impulsa je određena jednačinom:

Pout = Pn/40.95%

gde je Pn sadržaj 12-bitnog registra. Ukoliko je on 0 ili 4095 ($FFF) izlazni signal je jednosmeran, a ako je 2048 ($800) signal je kvadratnog oblika.

$D404 (54276) — CR1
Kontrolni registar oscilatora 1.

7	6	5	4	3	2	1	0
NOISE	PULSE	SAW	TRIANGLE	TEST	RING	SYNC	GATE

Ukoliko je neki bit na jedinici, aktivirana je određena funkcija.
GATE — Logička kontrola generatora obvojnice. U trenutku kada GATE postane 1, aktivira se attack faza. Kada GATE postane 0, završava se sustain faza.

SYNC — Sinhronizacija osnovne frekvencije oscilatora 1 sa osnovnom frekvencijom oscilatora 3.

RING — Kada je ovaj bit postavljen na jedinicu, i izabran trougaoni talasni oblik na oscilatoru 1, na njegovom izlazu pojaviće se proizvod signala iz oscilatora 1 i oscilatora 3, koji se zove ring modulisani signal. Pri tome se generišu neharmonijske frekvencije što je zgodno za imitiranje zvona, gonga ili za kreiranje specijalnih efekata.

TEST — Sve dok je ovaj bit na jedinici, oscilator 1 je na nuli što se često koristi u svrhe testiranja. Međutim, ova osobina može da se upotrebi i za sinhronizaciju oscilatora 1 sa spoljnim izvorom signala, generišući vrlo složene talasne oblike pod softverskom kontrolom.

TRIANGLE — Kada je na jedinici, na izlazu oscilatora 1 je trougaoni talasni oblik.

SAW — kada je na jedinici, na izlazu oscilatora 1 je testerasti talasni oblik.

PULSE — Kada je na jedinici, na izlazu oscilatora 1 je četvrtasti talasni oblik. Širina impulsa je određena sadržajem registra PWLO/PWHI.

NOISE — Kada je na jedinici, na izlazu oscilatora 1 se generiše šum. To je signal slučajne amplitude ali frekvencije oscilatora 1. Koristi se za imitiranje eksplozije, pucnja, vetra, bubnja itd.

$D405 (54277) ATTACK/DECAY
Ovaj registar zajedno sa registrom SUSTAIN/RELEASE, kontroliše rad generatora obvojnice za VCA koji je priključen direktno na izlaz oscilatora 1.
Biti 4 do 7 (ATKO-ATK3) određuju jednu od 16 brzina porasta signala u fazi ATTACK. Biti 0 do 3 (DCYO-DCY3) određuju jednu od 16 brzina opadanja signala u fazi DECAY.

$D406 (54278) SUSTAIN/RELEASE
Biti 4 do 7 (STNO-STN3) određuju jedan od 16 nivoa signala u odnosu na vršnu vrednost koja je bila u fazi ATTACK. Vrednost $F (15) daje maksimalnu amplitudu, a vrednost 0 daje signal amplitude 0. Treba uočiti da faza SUSTAIN traje sve dok je logički kontrolni signal na jedinici.
Biti 0 do 3 (RLSO-RLS3) određuju jednu od 16 brzina opadanja signala u fazi RELEASE. U trenutku kada logički kontrolni signal GATE postane nula, nivo signala počinje da opada zadatom brzinom od vrednosti određene u fazi SUSTAIN.
U sledećoj tabeli date su vrednosti trajanja pojedinih faza.

vrednost		ATTACK (ms)	DECAY/RELEASE (ms)
$0	(0)	2ms	6ms
$1	(1)	8ms	24ms
$2	(2)	16ms	48ms
$3	(3)	24ms	72ms
$4	(4)	38ms	114ms
$5	(5)	56ms	168ms
$6	(6)	68ms	204ms
$7	(7)	80ms	240ms
$8	(8)	100ms	300ms
$9	(9)	250ms	750ms
$A	(10)	500ms	1.5s
$B	(11)	800ms	2.4s
$C	(12)	1s	3s
$D	(13)	3s	9s
$E	(14)	5s	15s
$F	(15)	8s	24s

$D407 — $D40D (54279 — 54285) Registri glasa 2
Registri glasa 2 su funkcionalno identični glasu 1 ($D400 — $D406) sa sledećim izuzecima:
1. Kada se SYNC bit oscilatora 2 postavi na jedinicu, sinhronizuju se oscilator 1 i oscilator 2.
2. Kada se RING bit oscilatora 2 postavi na jedinicu i odabere trougaoni napon, na izlazu se dobija ring modulisani signal oscilatora 1 i oscilatora 2.

$D40E – $D414 (54286 – 54292) Registri glasa 3
Registri glasa 3 su funkcionalno identični registrima glasa 1 ($D400 – $D406) sa sledećim izuzecima:
1. Kada se SYNC bit oscilatora 3 postavi na jedinicu, sinhronizuju se oscilator 2 i oscilator 3.
2. Kada se RING bit oscilatora 3 postavi na jedinicu, i odabere trougaoni napon, na izlazu se dobija ring modulisani signal oscilatora 2 i oscilatora 3.

$D415 – $D416 ($54293 – 54294) FCLO/FCHI
Ova dva registra čine jedinstveni 11-bitni registar (biti 3 do 7 registra FCLO se ne koriste). Njegov sadržaj određuje graničnu frekvenciju niskopropusnog i visokopropusnog filtera ili centralnu (rezonantnu) frekvenciju filtera propusnika opsega. Opseg frekvencija je od 30Hz do 12KHz (pogledati poglavlje Hardver, 11.5 Audio kontroler).

$D417 (54295) RES/FILT
Biti 0 do 3 određuju koji će signal biti priključen na ulaz filtera.
FILT 1 (bit 0) Kada je na jedinici, signal iz glasa 1 dovodi se na ulaz filtera. Ako je na nuli, signal iz glasa 1 dovodi se na dalju obradu zaobilazeći filter.
FILT 2 (bit 1) Isto kao za FILT 1 samo za glas 2.
FILT 3 (bit 2) Isto kao za FILT 1 samo za glas 3.
FILTEX (bit 3) Isto kao za FILT 1 samo za spoljašnji audio signal.
Biti 4 do 7 (RES0 – RES3) određuju rezonancu (Q faktor) filtera. Vrednost 0 određuje minimalnu, a vrednost $F (15) maksimalnu rezonancu.

$D418 (54296) MODE/VOL
Biti od 0 do 3 (VOL0 – VOL3) određuju pojačanje izlaznog sabirnog pojačavača. Vrednost 0 daje pojačanje 0, a vrednost $F (15) maksimalno pojačanje.
Biti 4 do 7 određuju razne načine rada filtera:
LP (bit 4) Kada je na jedinici, filter je niskopropusni sa slabljenjem u propusnom opsegu od 12 dB/oktavi (filter drugog reda)
BP (bit 5) Kada je na jedinici, filter je propusnik opsega drugog reda.
HP (bit 6) Kada je na jedinici, filter je visokopropusni drugog reda.
3OFF (bit 7) Kada je na jedinici, glas 3 se isključuje iz puta audio signala. Postavljanjem bita FILT 3=0, sprečava se dolazak signala iz oscilatora 3 na izlazni pojačavač. Oscilator 3 tada može da radi kao LFO.
Načini rada filtera mogu se kombinovati. Na primer, moguće je istovremeno uključiti niskopropusni i visokopropusni filter kada se odgovarajućim podešavanjem njihovih graničnih frekvencija dobija filter nepropusnik opsega.

$D419 (54297) POTX
Ovo nije audio registar, ali se nalazi unutar audio kontrolera. U njemu je rezultat poslednje konverzije A/D konvertora 1 u opsegu od 0 do 255 (pogledati 11.5 Audio kontroler). Ova vrednost se obnavlja svakih 512 ciklusa takta Φ2.

$D41A (54298) POTY
Isto kao i za POTX samo važi za A/D konvertor 2.

$D41B (54299) OSC 3/RANDOM
Ovaj registar omogućuje mikroprocesoru očitavanje trenutne vrednosti amplitude na izlazu oscilatora 3. Pošto je signal oscilatora digitalno sintetizovan, njegova trenutna amplituda može da uzme vrednosti iz opsega od 0 do 255. Tako na primer, testerasti talasni oblik predstavlja niz rastućih brojeva od 0 do 255, a četvrtasti talasni oblik predstavlja skokovitu promenu sa 0 do 255 i nazad na 0. Ukoliko je odabran generator šuma, pojavljivaće se slučajni brojevi iz opsega 0 do 255. Pročitane vrednosti mogu se softverski menjati, a zatim koristiti za kontrolu ostalih elemenata sintesajzera. Tako se može u realnom vremenu menjati granična frekvencija niskopropusnog filtera ili se može proizvesti efekat zadržavanja trenutne vrednosti signala (engl. sample and hold).
Ukoliko se oscilator 3 koristi kao LFO, potrebno je postaviti FILT 3=0 i 3OFF=1 da se ne bi njegov signal pojavljivao na audio izlazu, već se koristio samo kao kontrolni napon.

$D41C (54300) ENV 3
Ovaj registar omogućuje mikroprocesoru očitavanje trenutne vrednosti amplitude kontrolnog napona na izlazu generatora obvojnice glasa 3. Početne vrednosti mogu se softverski menjati, a zatim koristiti za kontrolu ostalih elemenata sintesajzera. Tako se mogu generisati WAH-WAH ili „Phaser" efekti ako se dinamički kontrolišu granična frekvencija filtra ili širina četvrtastih impulsa respektivno. Treba napomenuti da je pre očitavanja neophodno startovati generator obvojnice postavljanjem bita GATE na jedinicu.

```
100 REM *** ZVUK1 ***
101 REM *************
102 :
103 SYS 8*4096
104 .OPT OO
105 SOUND = $D400
106 S = SOUND
107 VOICE3 = S+14
108 CTRL3 = S+18
109 PWHI = S+3
110 VOLUME = S+24
111 ADSR1 = S+6
112 CTRL1 = S+4
113 OSC3 = S+27
114 STOP = $FFE1
115 *= $7000
116 :
117 ; BRISANJE REGISTARA AUDIO KONTROLERA
118 ; ---------------------------------
119 :
120 :        LDA #0
121 :        LDX #24
122 :CLR     STA SOUND,X
123 :        DEX
124 :        BNE CLR
125 :        STA SOUND
126 :
127 ; INICIJALIZACIJA REGISTARA
128 ; ------------------------
129 :
130 :        LDA #5
131 :        STA VOICE3   ; DECAY3=5
132 :        LDA #16
133 :        STA CTRL3    ; OSC3=TROUGAONI SIGNAL
134 :        LDA #1
135 :        STA PWHI
136 :        LDA #143     ; JACINA NA MAX
137 :        STA VOLUME   ; OSC3 SE ISKLJUCUJE IZ AUDIO PUTANJE
138 :        LDA #%11111100
139 :        STA ADSR1    ; ADS NA MAX, R=0
140 :        LDA #%00100001; OSC1 TESTERASTI SIGNAL, GATE ON
141 :        STA CTRL1
142 ;
143 ; GENERISANJE ZVUKA
144 ; ---------------
145 :
146 :CONT    LDA #0
147 :        STA TEMP
148 :        LDA OSC3     ; CITANJE VRENOSTI NA IZLAZU OSC3
149 :        ASL A        ; I MNOZENJE SA OSAM
150 :        ROL TEMP
151 :        ASL A
152 :        ROL TEMP
153 :        ASL A
154 :        ROL TEMP
155 :        STA SOUND
156 :        CLC
157 :        LDA TEMP
158 :        ADC #$15     ; SABIRANJE OSNOVNOG KONTROLNOG NAPONA
159 :        STA SOUND+1  ; SA ONIM IZ LFO (OSC3)
160 :        JSR STOP     ; DA LI JE PRITISNUT TASTER STOP
```

```
161 :         BNE CONT     ; AKO NE, NASTAVI
162 :         LDA #0       ; AKO JESTE, ISKLJUCI ZVUK
163 :         STA VOLUME
164 :         RTS
165 TEMP - *
166 .END
```

```
100 REM *** CASOVNIK ***
110 REM ***************
120 :
130 SYS 8*4096
140 .OPT OO
150 SOUND - $D400
160 S - SOUND
170 VOICE1 - S+1
180 VOICE3 - S+15
190 VOLUME - S+24
200 ADSR1 - S+5
210 CTRL1 - S+4
220 *- $7000
230 ;
240 ; BRISANJE REGISTARA AUDIO KONTROLERA
250 ; -----------------------------------
260 :
270 :         LDA #0
280 :         LDX #24
290 :CLR      STA SOUND,X
300 :         DEX
310 :         BNE CLR
320 :         STA SOUND
330 ;
340 ; INICIJALIZACIJA REGISTARA
350 ; -------------------------
360 ;
370 :         LDA #130
380 :         STA VOICE1     ; FREKVENCIJA OSC1
390 :         LDA #9
400 :         STA ADSR1      ; A=0, D=9
410 :         LDA #30
420 :         STA VOICE3     ; FREKVENCIJA OSC3
430 :         LDA #15
440 :         STA VOLUME     ; JACINA NA MAX
450 ;
460 ; GENERISANJE ZVUKA
470 ; -----------------
480 :
490 :         LDX #12        ; BROJ OTKUCAJA
500 :COUNT    LDA #%00010101; OSC1 TROUGAONI SIGNAL,
510 :         STA CTRL1      ; GATE=1, RING=1
520 :         JSR DELAY
530 :         LDA #%00010100; GATE=0
540 :         STA CTRL1
550 :         JSR DELAY      ; KASNJENJE IZMEDJU OTKUCAJA
560 :         DEX            ; DA LI JE BIO POSLEDNJI OTKUCAJ
570 :         BNE COUNT      ; AKO NE, NASTAVI
580 :         LDA #0         ; AKO DA, ISKLJUCI ZVUK
590 :         STA VOLUME
600 :         RTS
610 :
```

```
620 ; POTPROGRAM ZA KASNJENJE
630 ; ----------------------
640 ;
650 ;DELAY  LDA #$FF
660 ;       STA TIME
670 ;       LDY #0
680 ;LOOP   NOP:NOP:NOP
690 ;       DEY
700 ;       BNE LOOP
710 ;       DEC TIME
720 ;       BNE LOOP
730 ;       RTS
740 ;TIME = *
750 .END
```

10
Grafika

Sve grafičke sposobnosti kojima Komodor raspolaže postoje zahvaljujući specijalizovanom video kontroleru 6567 (VIC 2).

Postoje tri osnovna načina rada VIC-a:

1. Rad sa karakterima
2. Rad u visokoj rezoluciji
3. Rad sa sprajtovima (pokretnim sličicama definisanim od strane korisnika).

Naravno, ovi načini mogu se međusobno kombinovati. Osim standardnog načina rada postoji i način rada sa povećanim brojem boja, ali sa dvostruko manjom rezolucijom.

Video kontroler pristupa memoriji u trenucima kada mikroprocesor ne obavlja nikakvu aktivnost na magistralama (za detalje pogledati poglavlje 11. Hardver). To omogućuje adresiranje memorijskih lokacija u celom adresnom prostoru računara. Pošto raspolaže sa 14 adresnih linija, moguće je adresirati memoriju u četiri segmenta po 16 Kbajta. Po uključenju računara to je nulti segment od $0000 do $3FFF. Za selekciju pojedinih segmenata korlste se linije PA0 i PA1 periferne jedinice CIA 2.

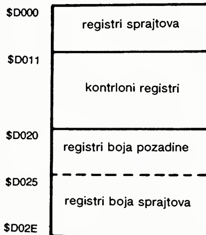
segment	PA1	PA0	adresni opseg
3	0	0	$C000 – $FFFF
2	0	1	$8000 – $BFFF
1	1	0	$4000 – $7FFF
0	1	1	$0000 – $3FFF

Komodorov bejzik ne podržava rad u visokoj rezoluciji niti rad sa sprajtovima. Zastupljen je samo standardni rad sa karakterima. Zbog toga se u ove svrhe programiranje mora obavljati na mašinskom jeziku.

10.1 REGISTRI VIC-a

Video kontroler ima ukupno 47 registara koji se mogu grupisati u tri celine:

– Kontrolni registri
– Registri boja
– Registri sprajtova



Sl. 10. 1. Organizacija registara VIC-a

10.1.1 Kontrolni registri

$D011 Registar načina rada (A)

7	6	5	4	3	2	1	0
RC8	ECM	EMM	DEN	RSEL	Y2	Y1	Y0

RC8 – osmi bit raster registra

ECM (engl. extended color mode) – Kada je ovaj bit na jedinici, moguće je koristiti pro-šireni skup boja pozadine karaktera. One mogu biti, u opštem slučaju, za svaki karakter različite.

BMM (engl. bit map mode) – Uključivanje rada u visokoj rezoluciji (320 x 200 tačaka). Pri tome svakoj tački odgovara jedan bit u memoriji.

DEN (engl. display enable) – Kada je ovaj bit na nuli, ništa se ne prikazuje na ekranu. Ekran je ceo obojen bojom okvira (koja je određena sadržajem registra $D020). U ovom slučaju, video kontroler pristupa memoriji isključivo u fazi i sistemskog takta, ne usporavajući rad mikroprocesora koji tada radi maksimalnom brzinom.

RSEL (engl. row select) – Normalno se na ekranu ispisuje 25 redova sa po 40 karaktera u redu. Postavljanjem ovog bita na nulu može se dobiti 24 reda, dok jedinica obezbeđuje 25 redova.

Y2 – Y0 (engl. Y scroll) – Standardno vertikalno pomeranje sadržaja ekrana je skokovito jer se obavlja za veličinu celog karaktera (8 raster linija). Ovo pomeranje može biti i sa manjim korakom (skoro kontinualno). Sadržaj Y2, Y1 i Y0 određuje za koliko raster linija će se izvršiti pomeranje.

$D012 Raster registar
Ovo su u suštini dva registra. Izbor jednog od njih vrši se linijom R/$\overline{W}$. Prema tome, prvi registar može samo da se čita. On daje (u kombinaciji sa RC8 iz registra $D011 koji je MSB) trenutnu raster poziciju. To je redni broj linije koja se pod kontrolom video kontrolera

ispisuje na ekranu. Da ne bi došlo do treperenja slike pri očitavanju sadržaja ovog registra, potrebno je ovo obaviti van vidljivog rasterskog opsega. Vidljivi opseg je u intervalu $033 – $0FB (51 – 251) linije.
Ukoliko se izvrši upisivanje u ovaj registar (uključujući i bit RC8), ova vrednost se pamti, a zatim upoređuje sa trenutnim brojem linije. Kada ova dva broja postanu jednaka, generiše se zahtev za prekidom (engl. raster interrupt).

$D013 LPX registar

$D014 LPY registar
Ova dva registra sadrže trenutnu X i Y poziciju svetlosne olovke. U trenutku nailaska mlaza na mesto gde se nalazi svetlosna olovka, generiše se impuls čija silazna ivica omogućuje punjenje ova dva registra. Pošto je interni brojač tačaka u VIC-u devetobitni, LPX može da registruje samo svaku drugu tačku. Interni brojač linija je osmobitni pa LPY registruje svaku tačku. Obnavljanje sadržaja ovih registara može se obaviti najviše jedanput unutar jedne slike (tj. svakih 1/25 sekunde).

$D015 Uključivanje sprajtova
Na ekranu može istovremeno da bude prisutno maksimalno osam sprajtova. Oni se nezavisno mogu uključivati i isključivati. Svakom sprajtu je dodeljen jedan bit iz ovog registra (sprajtu 0 – bit 0, sprajtu 1 – bit 1 itd. sve do sprajta 7 tj. bita 7). Ukoliko je dodeljeni bit na jedinici, odgovarajući sprajt je uključen, a ako je na nuli, odgovarajući sprajt je isključen.

$D016
Registar načina rada (B)

7	6	5	4	3	2	1	0
–	–	RES	MCM	CSEL	X2	X1	X0

Biti 7 i 6 se ne koriste.
RES Uvek je 0.
MCM (engl. multicolor mode) Ovaj način rada omogućuje ispisivanje u više boja unutar jednog karakter bloka (8 × 8 bita) ali sa dvostruko manjom rezolucijom. Aktivira se sa MCM = 1.
CSEL (engl. column select). Kada je ovaj bit na jedinici, ispisuje se 40 karaktera u redu, a kada je na nuli, 38 karaktera.
X2 – X0 (engl. X scroll) Sve napomene za Y2 – Y0 važe i ovde, samo se u ovom slučaju odnose na horizontalno pomeranje.

$D012 Registar vertikalnog povećanja sprajtova
Svakom od osam mogućih sprajtova dodeljen je jedan bit iz ovog registra (bit 0 za sprajt 0 ... i bit 7 za sprajt 7). Kada je on na jedinici, odgovarajući sprajt je povećan dva puta po vertikali. Pri ovome se rezolucija ne povećava već se samo udvostručuje veličina tačaka u smeru Y ose.

$D018
Pokazivači memorije

7	6	5	4	3	2	1	0
VM13	VM12	VM11	VM10	CB13	CB12	CB11	–

VM13 – VM10 Adresa video matrice (ekranske memorije) u okviru odabranog segmenta izražena u kilobajtima. Po uključenju računara, početna adresa je 0001 binarno, tj. video matrica počinje od lokacije $400 (1024). Ako se video memorija premešta, treba i ekranskom editoru staviti do znanja gde se ona nalazi. To se postiže postavljanjem nove vrednosti sistemske promenljive HIBASE.

CB13 – CB11 Adresa baze karaktera (videti poglavlje 11. Hardver), tj. karakter generatora. Ova adresa se menja u skokovima od po dva kilobajta u okviru odabranog segmenta.

SD019 Registar prekida

7	6	5	4	3	2	1	0
IRQ	–	–	–	ILP	IMMC	IMBC	IRST

IRQ (engl. interrupt request) Postoje četiri izvora prekida u VIC-u. Ukoliko nastupi bilo koji od njih, koji je dozvoljen, ovaj bit se postavlja na jedinicu. To dovodi do toga da se na izvodu 8 ($\overline{IRQ}$) pojavi nula zahtevajući od mikroprocesora prekid. Program za obradu prekida treba da odredi koji izvor je izazvao prekid.

ILP (engl. light pen interrupt). Postavlja se na jedinicu pri prolasku elektronskog mlaza preko mesta na ekranu gde se nalazi svetlosna olovka.

IMMC (engl. MOB-MOB collision) Postavlja se na jedinicu pri međusobnom sudaru dva sprajta i to samo pri prvom sudaru.

IMDC (engl. MOB-DATA collision) Postavlja se na jedinicu pri sudaru sprajta i nekog drugog objekta na ekranu (ivica ekrana, karakteri itd.). Ovo važi samo za prvi sudar.

IRST (engl. raster interrupt) Postavlja se na jedinicu kada se sadržaj internog brojača linija izjednači sa sadržajem raster registra.
Prilikom čitanja iz registra prekida, njegov sadržaj se ne briše.

SD01A Registar maske prekida.

7	6	5	4	3	2	1	0
–	–	–	–	ELP	EMMC	EMDC	ERST

Sva četiri izvora prekida mogu se selektivno maskirati (zabraniti) čime se sprečava generisanje zahteva za prekidom od dotičnog izvora.

ELP Prekid ILP: 1 = dozvoljen, 0 = zabranjen
EMMC Prekid IMMC: 1 = dozvoljen, 0 = zabranjen
EMDC Prekid IMDC: 1 = dozvoljen, 0 = zabranjen
ERST prekid IRST: 1 = dozvoljen, 0 = zabranjen

Registri za kontrolu sprajtova

	7	6	5	4	3	2	1	0	
SD01B	M7DP	M6DP	M5DP	M4DP	M3DP	M2DP	M1DP	M0DP	registar prioriteta
SD01C	M7MC	M6MC	M5MC	M4MC	M3MC	M2MC	M1MC	M0MC	izbor rada u više boja
SD01D	M7XE	M6XE	M5XE	M4XE	M3XE	M2XE	M1XE	M0XE	horizontalno povecanje
SD01F	M7M	M6M	M5M	M4M	M3M	M2M	M1M	M0M	sudar dva sprajta
SD01F	M7D	M6D	M5D	M4D	M3D	M2D	M1D	M0D	sudar sprajta sa pozadinom

— Registar prioriteta određuje da li će odgovarajući sprajt biti prikazan ispred ili iza elementa na ekranu. Svakom sprajtu odgovara jedan bit ovog registra. Kada je on nula, sprajt će biti prikazan ispred tj. zakloniće element na ekranu. Ako je neki bit na nuli, njemu dodeljen sprajt biće prikazan iza, tj. biće zaklonjen elementom na ekranu.

— Izbor rada u više boja. Svaki sprajt može se definisati kao višebojni, ako se odgovarajući bit ovog registra postavi na jedinicu. Pri tome je rezolucija upola manja.

— Horizontalno povećanje. Svaki sprajt može se dvostruko proširiti u pravcu X ose, pri čemu se ne povećava rezolucija. Dolazi samo do povećanja tačaka u horizontalnom pravcu.

— Sudar dva sprajta. Između sprajtova jednakog prioriteta može doći do sudara u vidljivom ili nevidljivom delu ekrana. Odgovarajući biti se tada postavljaju na jedinicu i to za sve sprajtove koji su učestvovali u sudaru. Takođe se, ukoliko je dozvoljeno, javlja zahtev za prekidom IMMC. Rutina za obradu prekida čitanjem ovog registra određuje između kojih sprajtova je došlo do sudara.
Sadržaj registra se briše pri čitanju.

— Sudar sprajtova sa pozadinom. Ukoliko dođe do sudara sprajta sa nekim karakterom ili elementom ekrana visoke rezolucije, odgovarajući bit se postavlja na jedinicu. Istovremeno se, ukoliko je dozvoljen, javlja zahtev za prekidom IMDC. Rutina za obradu prekida, čitanjem ovog registra, određuje koji sprajt se sudario sa elementom ekrana. Pri čitanju se briše sadržaj registra. Sudar može da se dogodi i van vidljivog dela ekrana.

10.1.2 Registri sprajtova

	7 … 0	sprajt
$D000	X0	
$D001	Y0	
$D002	X1	
$D003	Y1	
$D004	X2	
$D005	Y2	
$D006	X3	
$D007	Y3	
$D008	X4	
$D009	Y4	
$D00A	X5	
$D00B	Y5	
$D00C	Y6	
$D00D	Y6	
$D00E	X7	
$D00F	Y7	
$D010		XMSB

Svakom sprajtu dodeljen je par registara plus po jedan bit iz XMSB registra. Oni određuju položaj gornjeg levog ugla bloka unutar koga je definisan sprajt. U horizontalnom

pravcu postoji 512 mogućih položaja adresiranih sa 8 bita iz X registra dotičnog sprajta plus odgovarajući bit iz XMSB registra koji čini deveti bit. Pri tome su vidljivi samo položaji od $17 do $157 (23 do 347). I u vertikalnom pravcu svi položaji nisu vidljivi. Ima ih ukupno 256 (određeni su sa 8 bita Y registra), a vidljivi su u opsegu $32 — $F9 (50 — 249).

10.1.3 **Registri boja**

	7	6	5	4	3	2	1	0	
$D020	–	–	–	–	EC3	EC2	EC1	EC0	boja okvira
$D021	–	–	–	–	B0C3	B0C2	B0C1	B0C0	boja pozadine # 0
$D022	–	–	–	–	B1C3	B1C2	B1C1	B1C0	boja pozadine # 1
$D023	–	–	–	–	B2C3	B2C2	B2C1	B2C0	boja pozadine # 2
$D024	–	–	–	–	B3C3	B3C2	B3C1	B3C0	boja pozadine # 3

Ovi registri imaju po četiri bita što daje ukupno 16 različitih boja. Boja pozadine #0 se koristi u standardnom načinu rada dok se boje pozadine #1 do #3 koriste pri radu sa višebojnom pozadinom.

Registri boja sprajtova

	7	3	0	
$D025				višebojni sprajt # 0
$D026				višebojni sprajt # 1
$D027				boja 0. sprajta
$D028				boja 1. sprajta
$D029				boja 2. sprajta
$D02A				boja 3. sprajta
$D02B				boja 4. sprajta
$D02C				boja 5. sprajta
$D02D				boja 6. sprajta
$D02E				boja 7. sprajta

Ovi registri imaju po četiri bita što daje ukupno 16 različitih boja. Prva dva registra određuju boje sprajtova u višebojnom načinu prikazivanja. Ostali registri određuju boju pojedinih sprajtova. U ovom slučaju, svi sprajtovi mogu biti nezavisno obojeni.

10.2 **RAD SA KARAKTERIMA**

10.2.1 **Standardni karakteri**

Standardni rad sa karakterima je način koji Komodor koristi odmah po uključivanju. Aktivira se postavljanjem bita ECM, BMM i MCM, iz registra načina rada, na nulu. Pri tome su na ekranu prisutne tri boje: boja karaktera, boja pozadine i boja okvira. Svaki karakter

pojedinačno može biti jedne od 16 različitih boja, dok je pozadina jednobojna (takođe jedna od 16 boja). Okvir je jednobojan i u opštem slučaju je različite boje u odnosu na pozadinu.

Video kontroler uzima ekranski kôd iz ekranske memorije (video matrice) i dodaje ga na tzv. karakter bazu (tj. adresu karakter memorije). Na osnovu ovako formirane adrese, iz karakter memorije se čita sukcesivnih 8 bajtova. Pošto svaki ima po 8 bita, to čini matricu od 8 x 8 tačaka koje se ispisuju na ekranu.

Video matrica je organizovana kao matrica 40 x 25 karakter blokova (bajtova). Svakom bloku odgovara po jedan nibl (4 bita) u kolor memoriji koja počinje na adresi $D800 (55296). Zavisno od sadržaja pojedinog nibla, odgovarajući karakter biće ispisan jednom od 16 boja.

Boja pozadine je određena sadržajem registra $D021, a boja okvira, sadržajem registra $D020 video kontrolera.

Adresa kolor memorije je fiksna ali se adrese ekranske memorije i karakter memorije mogu menjati. Ekranska memorija mora da se nalazi unutar segmenta od 16KB sa kojim radi video kontroler. Ovo važi i za karakter memoriju. Adresa i jedne i druge memorije je određena sadržajem pokazivača memorije $D018. Po uključenju računara, ekranska memorija je na $400 (1024). Karakter memorija je ROM sa sledećim sadržajem:

	$D000 – $D1FF	velika slova
blok 0	$D100 – $D3FF	grafički karakteri
	$D400 – $D5FF	inverzna velika slova
	$D600 – $D7FF	inverzni grafički karakteri
	$D800 – $D9FF	mala slova
blok 1	$DA00 – $DBFF	velika slova i grafički simboli
	$DC00 – $DDFF	inverzna mala slova
	$DE00 – $DFFF	inverzna mala slova i grafički karakteri

Video kontroler može da radi samo sa jednim blokom od 2KB iz karakter ROM-a. Ovaj blok se bira izmenom bita CB11 u registru $D018. To je ekvivalentno istovremenom pritiskivanju tastera C= i SHIFT.

Karakter ROM se nalazi na istim adresama na kojima su i registri VIC-a. Međutim, ovde nikada ne dolazi do kolizije jer ROM-u pristupa samo video kontroler, a registrima samo mikroprocesor.

Po uključenju računara, video kontroler radi sa segmentom 0 ($0000 do $3FFF). Očigledno je da se karakter ROM ne nalazi u ovom opsegu. Međutim, video kontroler ipak može da mu pristupi zahvaljujući tome što je njegova adresa nepotpuno dekodovana. Na taj način se javljaju slike (ili senke) ROM-a i to na lokacijama $1000 – $1FFF (za segment 0) i $9000 – $9FFF (za segment 2). U segmentima 1 ($4000 – $8FFF) i 3 ($C000 – $FFFF) ne postoje senke karakter ROM-a pa se on u ovim slučajevima ne može koristiti. Ukoliko je neophodno da mikroprocesor pristupi karakter ROM-u radi čitanja njegovog sadržaja, potrebno ga je uključiti u adresni prostor pomoću linije $\overline{CHAREN}$ (bit 2 u internom registru mikroprocesora na adresi $0001). Na taj način moguće je kopirati sadržaj ROM-a u RAM, a zatim izvršiti eventualne izmene ostvarujući na taj način sopstveni skup karaktera. Naravno, karakter memorija je sada RAM pa informaciju o njenoj adresi treba uneti u registar $D018 video kontrolera. U sledećem primeru je pokazano kako se korišćenjem ovog principa može definisati karakter ž umesto karaktera £.

```
10 SYS 8*4096
20 .OPT OO
30 SYST = $01
40 ALO = $FB:AHI = $FC
50 BLO = $FD
60 BHI = $FE
70 SCRP = $0288
80 NMILO = $0318
90 NMIHI = $0319
100 WARM = $A002
110 CHAR = $D018
120 BLOK = $DD00
130 PREK = $DD0D
140 INVIC = $E518
150 CLS = $E544
160 FLAGS = $F6BC
170 INIO = $FDA3
180 NMIRS = $FE72
190 STOP = $FFE1
200 *= $C800
210 ;
211 ; PREMESTANJE KARAKTERA U RAM
212 ; ---------------------------
213 ;
220 :MOVE    SEI
230 :        LDA #$31
240 :        STA SYST
250 :        LDY #$00
260 :        STY ALO
270 :        STY BLO
280 :        LDA #$D0
290 :        STA AHI
300 :        LDA #$F0
310 :        STA BHI
320 :        LDX #$10
330 :PRENOS LDA (ALO),Y
340 :        STA (BLO),Y
350 :        DEY
360 :        BNE PRENOS
370 :        INC AHI
380 :        INC BHI
390 :        DEX
400 :        BNE PRENOS
410 :        LDY #$07
420 ;
430 ; DODAVANJE KVACICA
440 ; -----------------
490 :LOOP    LDA DAT1,Y
500 :        STA $F0E0,Y
510 :        STA $FB48,Y
570 :        LDA $F800,Y
580 :        STA $F8E0,Y
590 :        DEY
600 :        BPL LOOP
610 :        LDA DAT
640 :        STA $F8E1
650 :        LDA #$37
660 :        STA SYST
670 :        CLI
680 ;
682 ; POSTAVLJANJE PARAMETARA VIC-A
683 ; -----------------------------
684 ;
690 :SET     LDA #$94
700 :        STA BLOK
710 :        LDA #$CC
720 :        STA SCRP
730 :        LDA #$3C
740 :        STA CHAR
750 :        LDA #<NEWNMI
760 :        STA NMILO
770 :        LDA #>NEWNMI
780 :        STA NMIHI
790 :        JMP CLS
800 ;
801 ; PODACI ZA KVACICE
802 ; -----------------.
803 ;
810 :DAT    .BYTE $66,$3C,$66,$60,$60,$66,$3C,$00
820 :       .BYTE $66,$3C,$60,$3C,$06,$66,$3C,$00
830 :DAT1   .BYTE $66,$7E,$0C,$18,$30,$60,$7E,$00
840 ;
841 ; IZMENA VEKTORA NMI RUTINE
842 ; ------------------------
843 ;
850 :NEWNMI PHA
860 :        TXA
870 :        PHA
880 :        TYA
890 :        PHA
900 :        LDA #$77
910 :        STA PREK
920 :        LDY PREK
930 :        BMI RSNMI
940 :        JSR FLAGS
950 :        JSR STOP
960 :        BNE RSNMI
970 :        JSR INIO
980 :        JSR INVIC
990 :        JSR SET
1000 :       JMP (WARM)
1010 :RSNMI  JMP NMIRS
1020 .END
```

10.2.2 Višebojni karakteri

Način rada sa višebojnim karakterima omogućuje ispisivanje karaktera u četiri različite boje unutar karakter bloka, ali sa smanjenom rezolucijom. Aktivira se postavljanjem MCM na jedinicu, dok ECM i BMM ostaju na nuli. Da bi se svaka tačka unutar karaktera mogla obojiti jednom od četiri boje, potrebna su dva bita za njen opis. Zbog toga je „širina" tačke u horizontalnom smeru dva piksela, pa je nova matrica (karakter blok) veličine 4×8 tačaka. Kombinacije ova dva bita imaju sledeće značenje:

00 boja pozadine 0 – registar $D021
01 boja pozadine 1 – registar $D022
10 boja pozadine 2 – registar $D023
11 boja iz kolor memorije

Ukoliko se koriste prve tri kombinacije, promenom sadržaja odgovarajućeg registra moguće je istovremeno promeniti boju svih tačaka istih boja u neku drugu boju. Ako se koristi boja iz kolor memorije, sadržaj niblova ima sledeće značenje:

– Ako je standardni način rada sa karakterima, postoji 16 boja (0–15).

– Ako je višebojni način rada sa karakterima, postoje boje od 0 do 7 koje se prikazuju kao standardne tačke ako je odgovarajući karakter jednobojan. Ukoliko je višebojan, treba postaviti najteži bit u niblu na 1, a preostala tri bita će određivati boje (od 0 do 7) višebojnih tačaka. Prema tome, ako je npr. u neku lokaciju kolor memorije upisan broj 12, to će biti boja broj 4 u višebojnom načinu rada. Sledeći primer pokazuje kako se definiše karakter A koji će biti prikazan u četiri boje:

```
00011000    01100110
00111100    01100110
01100110    01100110
01111110    00000000
```

10.2.3 Višebojna pozadina

Način rada identičan je standardnim karakterima ali boja pozadine u okviru jednog karaktera može da bude jedna od četiri različitih boja. Sa druge strane, mogu biti prikazani samo oni karakteri sa ekranskim kodom od 0 do 63. Ovaj način rada se aktivira postavljanjem ECM bita na 1 dok su MCM i BMM na nuli. Nikada ne treba aktivirati istovremeno višebojne karaktere i višebojnu pozadinu. Boja pozadine se bira jednostavnim upisivanjem ekranskog koda u ekransku memoriju na osnovu sledeće tabele:

ekranski kod	boja pozadine je određena sadržajem registra broj	adresa
0–63	0	$D021
64–127	1	$D022
128–191	2	$D023
192–255	3	$D024

Tako na primer, ekranski kod 1 daće slovo A sa bojom pozadine određenom sadržajem registra $D021 dok će kôd 65 dati takođe slovo A ali sa bojom pozadine određenom sadržajem registra $D022. Boja samog karaktera, određena je sadržajem odgovarajućeg nibla kolor memorije, tj. može biti jedna iz palete od 16 boja.

10.3 RAD U VISOKOJ REZOLUCIJI

10.3.1 Standardni način rada

Kod rada u visokoj rezoluciji moguće je ispisati 320×200 tačaka na ekranu. Svakoj tački odgovara po jedan bit u memoriji. Ukoliko je bit na jedinici, tačka je boje ispisa (engl. fereground color), a ukoliko je na nuli, tačka je boje pozadine (engl. background color). Pošto ima ukupno 64000 tačaka, za njihovo smeštanje potrebno je nešto manje od 8 kilobajta.

Način rada u visokoj rezoluciji se aktivira postavljanjem bita BMM registra $D011 na jedinicu. Pri tome je MCM=0. Video kontroler sada ne čita podatke iz ekranske memorije, već iz jednog od dva bloka od 8KB unutar memorijskog segmenta od 16KB sa kojim radi. Postoji ukupno 8 ovakvih blokova unutar adresnog prostora od 64KB. Koji će od njih biti odabran zavisi od odabranog segmenta memorije (PA0 i PA1 jedinice CIA 2) kao i od vrednosti bita CB13 registra $D018 (0=blok 0, 1=blok 1).

Moguće je, u opštem slučaju, odabrati bilo koji blok, ali su dva praktično neupotrebljiva zbog već napomenutog postojanja slike karakter ROM-a usled nepotpunog dekodovanja njegove adrese. To su blok 0 u segmentu 0 ($0000 — $1FFF) i blok 0 u segmentu 2 ($8000 — $9FFF). Prema tome, ostaju na raspolaganju sledeći blokovi:

blok	segment
1	0
0	1
1	1

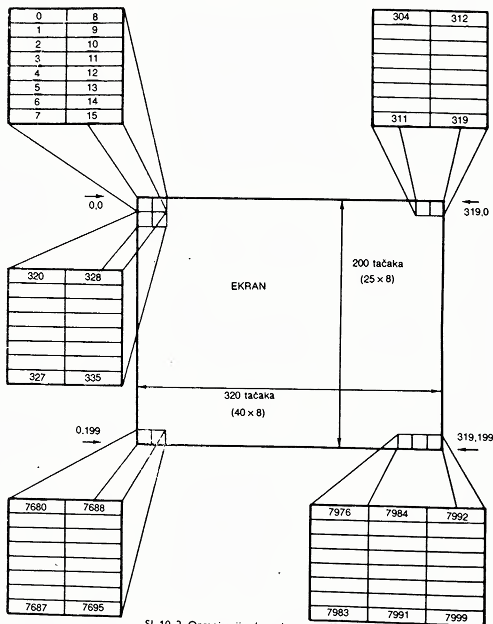
Korišćenjem jednog od ovih blokova smanjuje se veličina raspoložive memorije za smeštaj bejzik programa pa se najčešće koriste sledeći blokovi:

blok	segment	napomena
1	2	prostor bejzik ROM-a
1	3	prostor Kernal ROM-a

Na osnovu organizacije memorije (videti poglavlje 8) vidi se da se na adresama na kojima su bejzik i Kernal ROM, nalazi i RAM. Kada mikroprocesor čita sa ovih lokacija, on čita sadržaj ROM-a, a kada upisuje, on upisuje u RAM „iza" ROM-a. Video kontroler čita sadržaj ovog RAM-a samo u trenucima kada mikroprocesor nema nikakvih aktivnosti na magistralama. Zbog toga je nemoguće u ovim blokovima smeštati programe ili podatke već samo sliku koju će preuzimati video kontroler. Naravno, isključivanjem bejzik i Kernal ROM-a (pomoću linija $\overline{LORAM}$ i $\overline{HIRAM}$), ovaj RAM postaje standardno upotrebljiv. Blok 0 segmenta 3 se ne koristi jer se u njemu nalaze registri ulazno izlaznih jedinica.

Ekranska memorija se i dalje koristi ali je njena funkcija izmenjena. Svaki bajt ekranske memorije podeljen je na dva nibla. Nibl manje težine određuje jednu od 16 boja pozadine, dok nibl veće težine određuje jednu od 16 boja ispisa. Prema tome, u ovom načinu rada, u svakom karakter bloku (matrica tačaka 8×8) mogu postojati dve boje. Ove se boje, u opštem slučaju, mogu razlikovati među različitim karakter blokovima.

Kolor memorija se u ovom načinu rada ne koristi.



Sl. 10. 2. Organizacija ekranske memorije

Bajtovi su unutar bloka organizovani kao što je prikazano na slici 10.2. Brojevi unutar pravougaonika su redni brojevi bajtova u odnosu na početak bloka.

Pošto svakoj tački ekrana odgovara po jedan bit unutar odabranog bloka, ovakav način prikazivanja naziva se bitna mapa (bitni zapis) ekrana, dok se ovakav način rada u visokoj rezoluciji često naziva, bit mapirani.

Početak bloka je istovremeno i koordinatni početak. Koordinata X uzima vrednosti od 0 do 319, a koordinata Y od 0 do 199.

Da bi se uspostavila korespondencija između koordinata neke proizvoljne tačke A(x, y) i adrese odgovarajućeg bita u memoriji, potrebno je koristiti sledeće formule:

Red u kome se nalazi matrica 8×8 unutar koje je tražena tačka:

RED=INT (Y/8)

Početna adresa reda:

ADRESA=OSNOVA+RED * 320

gde je OSNOVA početna adresa odabranog bloka.

Redni broj (u okviru izračunatog reda) kvadrata (matrice 8×8) u kome je tačka:

KVADRAT=ADRESA+8 * INT (X/8)

Adresa bajta unutar koga je tačka:

BAJT=KVADRAT+Y AND 7

Redni broj bita koji je adresiran (unutar prethodno određenog bajta):

BIT=7−(X AND 7)

Sređivanjem gornjih jednačina dobija se:

BAJT=OSNOVA+INT (Y/8) * 320+8 * INT (X/8)+Y AND 7
BIT=7−(X AND 7)

Postavljanje željenog bita na jedinicu:

(BAJT)=(BAJT) OR (2↑BIT)

Postavljanje željenog bita na nulu:

(BAJT)=(BAJT) AND (255−2 ↑BIT)

(BAJT) označava sadržaj lokacije sa adresom BAJT.

Na osnovu prethodnih jednačina, moguće je raditi u visokoj rezoluciji koristeći bilo koji programski jezik (pa i bejzik). U sledećem primeru dat je program na mašinskom jeziku koji ovo i ilustruje i koji se sastoji od pet delova:

INIT — uključivanje rada u visokoj rezoluciji
CLEAR — brisanje sadržaja ekrana u visokoj rezoluciji
COLOR — postavljanje boje pozadine
RESET — isključivanje i uključivanje tačke sa koordinatom x, y
SWOFF — povratak u standardni način rada sa karakterima

Ovi delovi su organizovani kao potprogrami tako da se mogu pozivati iz bejzika. Pri tome su korišćene neke standardne rutine bejzik interpretera i Kernala (za detalje pogledati poglavlje 8) pomoću kojih se prenose parametri.

Na kraju je dat bejzik program koji demonstrira korišćenje opisanih mašinskih rutina.

```
100 SYS 8*4096
104 .OPT OO
108 *= $C000
112 XCOORO = $14
116 SECADR = $B9
120 TEMP = $FO
124 AORESS = TEMP
128 COLORL = $0400
132 COLORH = $0800
136 GRAPHL = $2000
140 GRAPHH = $4000
144 CHKCOM = $AEFO
148 GETBYT = $B79E
152 GETCOO = $B7EB
156 GETPAR = $E104
160 BSOUT = $FFD2
172 VIOEO = $D000
176 FALSE = 255
```

```
180 TRUE = 0
184 CLS = 128+19
188 *= $C000
191 ;
192 ; TABELA SKOKOVA
193 ; -------------
196 :         JMP INIT
200 :         JMP CLEAR
204 :         JMP COLOR
208 :         JMP SET
212 :         JMP RESET
216 :         JMP SWOFF
217 ;
228 ; INICIJALIZACIJA VISOKE REZOLUCIJE
225 ; --------------------------------
232 :INIT    LDA VIOEO+17
236 :         STA SCRATCH+1
240 :         LDA VIDEO+24
244 :         STA SCRATCH
248 :         LDA #27+32
252 :         STA VIDEO+17
256 :         LDA #16+8
260 :         STA VIOEO+24
264 :         LDX #16
268 :         JMP COLOR1
272 ;
273 ; BRISANJE EKRANA VISOKE REZOLUCIJE
274 ; --------------------------------
276 :CLEAR   LOY #0
280 :         LDA #>GRAPHL
284 :         STY TEMP
288 :         STA TEMP+1
292 :CLEAR1  TYA
296 :CLEAR2  STA (TEMP),Y
300 :         INY
304 :         BNE CLEAR2
308 :         INC TEMP+1
312 :         LOA TEMP+1
316 :         CMP #>GRAPHH
320 :         BNE CLEAR1
324 :         RTS
328 ;
329 ; POSTAVLJANJE BOJE POZADINE
330 ; -------------------------
332 :COLOR   JSR CHKCOM
336 :         JSR GETBYT
340 :COLOR1  LDY #0
344 :         LOA #>COLORL
348 :         STY TEMP
352 :         STA TEMP+1
356 :COLOR2  TXA
360 :COLOR3  STA (TEMP),Y
364 :         INY
368 :         BNE COLOR3
372 :         INC TEMP+1
376 :         LOA TEMP+1
380 :         CMP #>COLORH
384 :         BNE COLOR2
388 :OUTRAN  RTS
392 ;
393 ; BRISANJE/CRTANJE TACKE
```

```
394 ; ----------------------
396 :RESET  LDA #FALSE   ;BRISANJE
400 :       BNE SET1
404 :SET    LDA #TRUE    ;CRTANJE
408 :SET1   STA RSFLG
412 :       JSR CHKCOM
416 :       JSR GETCOO   ;CITANJE KOORDINATA
420 :       CPX #200     ;ISPITIVANJE DA LI SU KOORDINATE
424 :       BCS OUTRAN   ;UNUTAR OPSEGA 320 X 200
428 :       LDA XCOORD
432 :       CMP #<320
436 :       LDA XCOORO+1
440 :       SBC #>320
444 :       BCS OUTRAN
448 :       TXA
452 :       LSR          ;REO=INT(Y/8)
456 :       LSR
460 :       LSR
464 :       ASL
468 :       TAY
472 :       LDA MUL320,Y ;RED=RED*320
476 :       STA ADRESS
480 :       LDA MUL320+1,Y
484 :       STA ADRESS+1
488 :       TXA
492 :       AND #%00000111 ;Y=Y AND 7
496 :       CLC
500 :       AOC ADRESS   ; IZRACUNAVANJE IZRAZA ZA AORESIRANJE BAJTA
504 :       STA ADRESS
508 :       LDA ADRESS+1
512 :       ADC #0
516 :       STA ADRESS+1
520 :       LDA XCOORO
521 :       AND #%00000111
522 :       TAY
523 :       LOA XCOORD
524 :       ANO #%11111000
528 :       CLC
532 :       AOC ADRESS
536 :       STA ADRESS
540 :       LDA ADRESS+1
544 :       ADC XCOORO+1
548 :       STA AORESS+1
552 :       LOA ADRESS
556 :       CLC
560 :       ADC #<GRAPHL
564 :       STA ADRESS
568 :       LDA ADRESS+1
572 :       ADC #>GRAPHL
573 :       STA ADRESS+1
576 :       LDX #0
580 :       LDA (ADRESS,X)
584 :       BIT RSFLG
588 :       BPL SET2
592 :       AND ANDMAS,Y
596 :       JMP SET3
600 :SET2   ORA ORMASK,Y
604 :SET3   STA (ADRESS,X)
608 :       RTS
612 ;
613 ; POVRATAK U RAD SA KARAKTERIMA
```

```
614 ; ----------------------------
616 :SWOFF  LDA SCRATCH+1
620 :       STA VIDEO+17
624 :       LDA SCRATCH
628 :       STA VIDEO+24
632 :       LDA #CLS
636 :       JMP BSOUT
640 :N = 320
644 :MUL320 .WORD 0*N,1*N,2*N,3*N,4*N
648 :       .WORD 5*N,6*N,7*N,8*N,9*N
652 :       .WORD 10*N,11*N,12*N,13*N,14*N
656 :       .WORD 15*N,16*N,17*N,18*N,19*N
660 :       .WORD 20*N,21*N,22*N,23*N,24*N
664 :ORMASK .BYTE %10000000
668 :       .BYTE %01000000
672 :       .BYTE %00100000
676 :       .BYTE %00010000
680 :       .BYTE %00001000
684 :       .BYTE %00000100
688 :       .BYTE %00000010
692 :       .BYTE %00000001
696 ;
700 :ANDMAS .BYTE %01111111
704 :       .BYTE %10111111
708 :       .BYTE %11011111
712 :       .BYTE %11101111
716 :       .BYTE %11110111
720 :       .BYTE %11111011
724 :       .BYTE %11111101
728 :       .BYTE %11111110
732 SCRATCH .WORD 0
736 RSFLG   .WORD 0
740 YCOORD  .WORD 0
744 .END
999 :
1000 REM *** BEJZIK PROGRAM ***
1005 REM ********************
1006 :
1010 SYS 12*4096
1020 SYS 12*4096+3
1030 SYS 12*4096+6,1
1050 FOR X= 0 TO 319
1060 Y=200-100*(SIN(X*6.28/160)+1)
1070 SYS 12*4096+9,X,Y
1080 NEXT
1090 GOTO 1090
```

10.3.2 Višebojni način rada

Višebojni način rada se inicijalizuje isto kao i standardni način rada u visokoj rezoluciji. Pri tome su MCM=1 i BMM=1. Na ovaj način moguće je prikazati četiri nezavisne boje unutar svakog karakter bloka od 8×8 piksela. U ovom slučaju pojedine tačke zauzimaju po dva piksela što dovodi do smanjenja rezolucije na 160×200, što je slično slučaju višebojnih karaktera. Slika je određena bitnom mapom pri čemu su svakoj tački dodeljena dva bita (B0 i B1).

B0	B1	Boja tačke
0	0	boja pozadine 0 — registar $D021
0	1	boja određena gornjim niblom u ekranskoj memoriji
1	0	boja određena donjim niblom u ekranskoj memoriji
1	1	boja određena niblom u kolor memoriji

Da bi se uspostavila korespondencija između koordinata neke proizvoljne tačke A(x, y) i adrese bitnog para, potrebno je koristiti sledeće formule:
Red u kome se nalazi matrica 8 × 8 unutar koje je traženi par bita:

RED = INT (Y/8)

Početna adresa reda:

ADRESA = OSNOVA + RED * 320

gde je OSNOVA početna adresa odabranog bloka.

KVADRAT = ADRESA + 8 * INT (X/4)

Adresa bajta unutar koga je tačka:

BAJT = KVADRAT + Y AND 7

Sređivanjem se dobija:

BAJT = OSNOVA + INT (Y/8) * 320 + 8 * INT (X/4) + Y AND 7

Redni broj težeg bita bitnog para unutar prethodno određenog bajta:

BIT = 6 − 2 * (X AND 3)

Uključivanje adresirane tačke:

(BAJT) = (BAJT) AND (255 − 3 * 2 ↑BIT) OR (B1B0) * 2 BIT

Isključivanje adresirane tačke:

(BAJT) = (BAJT) AND (255 − 3 * 2 ↑BIT)

10.4 RAD SA SPRAJTOVIMA

Sprajtovi su pokretne sličice definisane od strane korisnika. Komodorov video kontroler podržava rad sa 8 nezavisnih sprajtova od kojih svaki može biti različite boje. Svaki sprajt ima i svoj definicioni blok, pokazivač, registar za x i y koordinatu, kolor registar, bit za uključivanje i bit za detekciju sudara.

10.4.1 Definisanje sprajtova

Sprajtovi se definišu kao i karakteri ali unutar matrice od 24 × 21 tačka. Za to je potrebno 63 bajta organizovanih kao na slici.

bajt 0	bajt 1	bajt 2
bajt 3	bajt 4	bajt 5
⋮	⋮	⋮
bajt 60	bajt 61	bajt 62

Sl. 10. 3. Organizacija bajtova pri definisanju sprajtova

Za standardne sprajtove važi sledeće:

Svakoj tački na ekranu odgovara jedan bit unutar bloka za definisanje dotičnog sprajta. Ukoliko je neki bit na jedinici, tačka na ekranu će biti boje ispisa. Ako je na nuli, tačka će biti boje pozadine. Svakom sprajtu je dodeljen jedan registar boje (4 bita) -- registri $D027 – $D02E.

Za višebojne sprajtove važi sledeće:

Svakoj tački na ekranu odgovaraju dva bita unutar bloka za definisanje sprajta. Na taj način se rezolucija sprajta smanjuje na 12×21 tačku.

Rad sa više boja postiže se postavljanjem odgovarajućeg bita registra $D01C na jedinicu. U tom slučaju kombinacije bita koje određuju boju sprajta su:

B1	B0	boja
0	0	transparentna
0	1	određena registrom $D025
1	0	određena registrima $D027 – $D02E (za svaki sprajt može da bude različita)
1	1	određena registrom $D026

Transparentna boja je boja pozadine ili boja elementa koji se nalazi iza posmatranog sprajta.

Već je rečeno da se ekranska memorija nalazi unutar bloka od 1KB (1024 bajta). Ona zauzima tačno 1000 bajtova. Od preostalih 24 bajta poslednjih 8 se koriste kao pokazivači svakog sprajta pojedinačno. Oni sadrže redni broj bloka od 64 bajta (63 za definiciju i jedan za razdvajanje) u odnosu na početak segmenta memorije sa kojom radi video kontroler. Blokova ima 256 pa se dobija 256 * 64=16KB tj. veličina celog segmenta. U prikazanom primeru pokazivač sprajta 0 (lokacija 2040) sadrži broj 13 što znači da je sprajt definisan sa početkom od lokacije 13 * 64=832.

10.4.2 **Uključivanje i pozicioniranje sprajtova**

Sprajtovi se uključuju postavljanjem na jedinicu odgovarajućeg bita u registru $D015. Svakom sprajtu odgovara jedan bit ovog registra. Sprajtovi se isključuju postavljanjem na nulu odgovarajućeg bita u registru $D015.

Svakom sprajtu je dodeljen jedan par registara sa x i y koordinatama gornjeg levog ugla pravougaonika unutar koga se on nalazi. Oba registra su osmobitna. U pravcu x koordinate ima više tačaka (320) od maksimalnog broja koji X registar može da adresira (256). Zbog toga je ovom registru dodeljen još jedan bit sa težinom 256 tako da je adresni opseg X registra sada 512 tačaka. Ovo je sada više od broja tačaka na ekranu pa su vidljive samo one tačke sa koordinatama između 24 i 343. Svakom od osam X registara dodat je po jedan bit. Svi biti su organizovani u jedan registar XMSB ($D010).

U pravcu y koordinate ima manje tačaka (200) od maksimalnog broja koji Y registar može da adresira pa su vidljive samo one sa koordinatama između 50 i 249.

Koordinatni početak je u gornjem levom uglu ekrana.

Svaki sprajt može se dvostruko proširiti u smeru x i y ose postavljanjem na jedinicu odgovarajućeg bita registra $D010 i $D017 respektivno. Pri tome se rezolucija ne povećava već svaka tačka zauzima dvostruko više piksela.

Prioritet prikazivanja sprajtova određen je registrom $D01B. Ukoliko je odgovarajući bit na jedinici, sprajt ima niži prioritet i u slučaju preklapanja koordinata sa elementima ekrana (karakter ili grafika) biće prikazan iza njega. Ukoliko je odgovarajući bit na nuli, sprajt je višeg prioriteta tj. biće prikazan ispred.

Sprajtovi međusobno imaju fiksiran prioritet pri čemu onaj sa rednim brojem 0 ima najviši, sa rednim brojem 1 niži i sve do onog sa rednim brojem 7 koji ima najniži prioritet.

```
5 REM *** SPRAJT ***
6 REM ***************
7 :
10 SYS 8*4096
20 .OPT 00
30 *- $C000
100 :        LDX #0
110 :NEXT    LDA DATA,X;UCITAVANJE PODATAKA U BLOK 13
120 :        STA 832,X
130 :        INX
140 :        CPX #63
150 :        BNE NEXT
151 :        LDA #13    ; POSTAVLJANJE POKAZIVACA NA BLOK 13
152 :        STA 2040
153 :        LDX #30    ;POCETNA X KOORDINATA
170 :LOOP    STX $D000 ;SPRAJT 0, X KOORDINATA
180 :        LDA #59
190 :        STA $D001 ;SPRAJT 0, Y KOORDINATA
200 :        LDA #1
210 :        STA $D015 ;UKLJUCENJE SPRAJTA
211 :        JSR DELAY
212 :        INX        ; PROMENA X KOORDINATE
213 :        CPX #200
214 :        BNE LOOP
220 :        RTS
221 :DELAY   LDY #255   ; POTPROGRAM ZA KASNJENJE
222 :L4      NOP:NOP:NOP:
223 :        DEY
224 :        BNE L4
225 :        RTS
227 ;
228 ; PODACI ZA DEFINISANJE SPRAJTA
229 ; ---------------------------
230 :DATA   .BYTE 255,0,255,128,0,1,128,0,1,128,0,1,128,0,1,128,0,1
240 :       .BYTE 128,0,1,0,8,0,0,8,0,0,8,0,0,54,0,0,8,0,0,8,0,0
250 :       .BYTE 8,0,128,0,1,128,0,1,128,0,1,128,0,1,128,0,1
260 :       .BYTE 128,0,1,255,0,255
270 .END
```

10.4.3 Sudari

Postoje dve vrste sudara: Sudar između sprajtova i sudar sprajta sa pozadinom.

Sudar između sprajtova nastaje kada se delovi sprajtova koji nisu transparentni nađu na istim koordinatama. U tom slučaju se u registru $D01E odgovarajući biti oba sprajta postavljaju na jedinicu i generiše se (ukoliko je dozvoljen) zahtev za prekidom.

Sudar između netransparentnog dela sprajta i pozadine (ovo je samo uslovno pozadina tj. to mogu biti karakteri i elementi grafike visoke rezolucije) manifestuje se postavljanjem odgovarajućeg bita registra $D01F na jedinicu i generisanjem (ukoliko je dozvoljeno) zahteva za prekidom.

10.5 MEŠOVITI NAČIN RADA

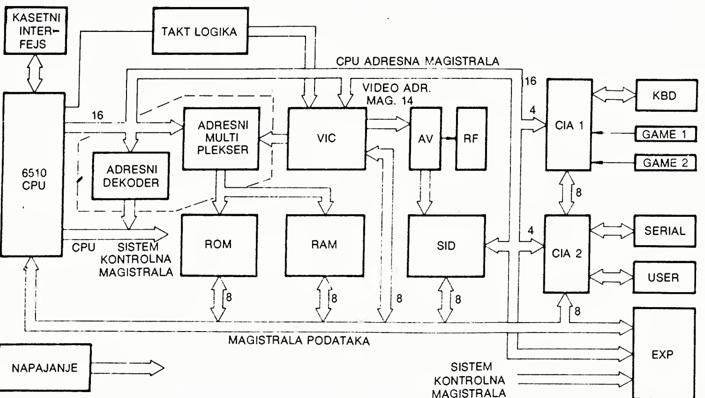
Svi načini rada međusobno se mogu mešati. Sprajtovi mogu biti prisutni na ekranu bez obzira na to da li se radi u visokoj rezoluciji ili sa karakterima. Ukoliko je potrebno da se istovremeno radi u visokoj rezoluciji i sa karakterima, primenjuje se tehnika rasterskih prekida. Isto se radi i kada je potrebno da jedan deo ekrana bude u visokoj rezoluciji, a drugi višebojni, kao i kada je potrebno istovremeno prisustvo više od 8 sprajtova.

Tehnika rasterskih prekida zasnovana je na upotrebi raster registra $D012. Rasterski prekid nastaje kada elektronski mlaz na ekranu ispiše određenu liniju. Koja je to linija određeno je sadržajem raster registra. Po nastanku prekida moguće je izmeniti način rada video kontrolera. Posle nastanka novog prekida može se vratiti u prethodni način rada itd. Broj linije kod koje će doći do prekida zadaje se upisivanjem u raster registar. Generisanje prekida može se zabraniti postavljanjem bita ERST na nulu u registru maske prekida $D01A.

U sledećem primeru prikazana je upotreba tehnike rasterskih prekida. U gornjem delu ekrana tekst će biti ispisan velikim slovima na crnoj pozadini dok će u donjem delu biti ispisan malim slovima na sivoj pozadini.

```
10 SYS 8*4096
20 .OPT OO
30 *= $C000
100 ;
110 ; IZMENA VEKTORA IRQ RUTINE
120 ; -----------------------
130 ;
140 :RASTER SEI
150 :       LDA #$7F
160 :       STA $DC0D
170 :       LDA #1
180 :       STA $D01A
190 :       LDA #146
200 :       STA $D012
210 :       LDA #27
220 :       STA $D011
230 :       LDA #<IRQ
240 :       STA $314
250 :       LDA #>IRQ
260 :       STA $315
270 :       CLI
280 :       RTS
290 ;
300 ; NOVA RUTINA ZA OBRADU PREKIDA
310 ; ---------------------------
320 ;
330 :IRQ    LDA $D012
340 :       CMP #146
350 :       BNE PRVI
355 ;
356 ; RUTINA ZA DRUGI DEO EKRANA
360 :       LDA #0
370 :       STA $D012
380 :       LDA #23
390 :       STA $D018
400 :       LDA #15
405 :       STA 53281
406 :       LDA #1
```

```
410 :        STA $D019
420 :        PLA
430 :        TAY
440 :        PLA
450 :        TAX
460 :        PLA
470 :        RTI
480 ;
490 ; RUTINA ZA PRVI DEO EKRANA
500 :PRVI    LDA #146
510 :        STA $D012
520 :        LDA #0
530 :        STA 53281
540 :        LDA #21
550 :        STA $D018
560 :        LDA #1
570 :        STA $D019
580 :        JMP $EA31   ;SKOK U STANDARDNU IRQ RUTINU
590 .END
```



Sl. 11. 1. Blok šema računara Commodore 64

11
Hardver

Među osmobitnim kućnim računarima Komodor 64 zahvaljujući svojoj dobroj hardverskoj podlozi, spada u one računare koji pružaju najviše raznovrsnih primena. Mogućnosti primene zavise isključivo od stepena povezanosti računara sa spoljnim svetom. Kod Komodora je, zahvaljujući dobrom balansiranju između hardvera i softvera, obezbeđena vrlo dobra povezanost računara sa perifernim uređajima preko većeg broja standardnih veza. To su:

- video
- audio
- TV
- tastatura
- upravljačka palica
- svetlosna olovka
- A/D konvertor
- IEEE 488
- RS 232
- Centronics

Elektronska kola Komodora se mogu podeliti u devet većih celina:

1. mikroprocesor
2. RAM
3. ROM
4. video kontroler
5. audio kontroler
6. periferne jedinice
7. kolo za upravljanje memorijom
8. kolo za generisanje taktova
9. napajanje

Na slici 11.1 prikazana je blok šema Komodorovog hardvera. Treba uočiti postojanje više sistemskih i lokalnih magistrala dok postoji samo jedna magistrala podataka. Adresnih magistrala ima tri i to su: CPU magistrala (sistemska), video magistrala (lokalna) i multipleksirana magistrala (lokalna). Takođe postoje dve kontrolne magistrale: CPU magistrala (lokalna) i sistem kontrola magistrala (sistemska).

Adresni multiplekser i adresni dekoder čine kolo za upravljanje memorijom (engl. memory management unit).

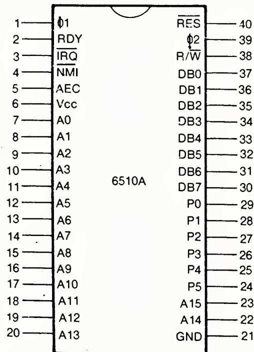
11.1 MIKROPROCESOR

Centralna procesorska jedinica u Komodoru 64 je osmobitni mikroprocesor 6510A. Razvijen je od strane firme MOS kao poboljšana varijanta čuvenog mikroprocesora 6502. 6510A je softverski potpuno kompatibilan sa ovim mikroprocesorom, dok postoje neke hardverske razlike.

Sve operacije, koje mikroprocesor obavlja, vremenski su sinhronizovane u odnosu na takt (engl. clock). Mikroprocesor 6510A zahteva upotrebu takozvanog dvofaznog takta. To su dva signala pravougaonog oblika, ali suprotnih faza. Kod Komodora frekvencija takta iznosi 0.98525MHz i dobija se iz logike za generisanje taktova.

Funkcije pojedinih izvoda mikroprocesora 6510A

Izvodi mikroprocesora mogu da budu ulazi ili izlazi, ili i jedno i drugo. Pojedini izlazi imaju mogućnost da pređu u stanje visoke impedanse, čime omogućavaju nekoj drugoj jedinici da upravlja stanjem na liniji (eng. three state). Iznad simbola pojedinih izvoda se nalazi crta koja označava da je za njega aktivno stanje (pri kome vrši svoju funkciju) logička nula, a normalno, neaktivno stanje je logička jedinica.



Sl. 11. 2. Raspored izvoda mikroprocesora 6510A

DB0 do DB7, magistrala podataka, dvosmerne linije (ulazi ili izlazi). Preko ovih osam izvoda se razmenjuju podaci između mikroprocesora i perifernih jedinica ili memorije.

A0 do A15, adresna magistrala, izlazi, imaju mogućnost da budu u stanju visoke impedanse. Preko ovih 16 izvoda mikroprocesor može da adresira 65536 (2^{16}) lokacija u memoriji.

P0 do P5, dvosmerne linije periferijskog registra (ulazi ili izlazi zavisno od sadržaja registra smera podataka). U adresnom prostoru periferijski registar je na adresi $0000 dok je registar smera podataka (engl. DDR — data direction register) na adresi $0001.

Φ1, takt faza 1 (engl. clock phase one), ulaz

Φ2, takt faza 2 (engl. clock phase two), izlaz

RDY, (engl. READY), ulaz

Koristi se za ubacivanje ciklusa čekanja (engl. wait cycles — videti vremenske dijagrame) pri radu sa sporim memorijama ili pri direktnom pristupu memoriji. Da bi se

ovi ciklusi ubacili, RDY ulaz mora u toku $\Phi 1$ impulsa takta da promeni vrednost sa logičke jedinice na logičku nulu. Ovo mora da se obavi u mašinskom ciklusu koji ne odgovara vremenu upisivanja.

$\overline{\text{RES}}$, reset, ulaz

Koristi se za resetovanje ili ponovno startovanje mikroprocesora po uključivanju napajanja. Dok je ova linija aktivna, razmena podataka sa mikroprocesorom je nemoguća. Kada se ovaj izvod dovede na logičku jedinicu, mikroprocesor počinje da izvršava reset sekvencu. Prvo propusti da prođu 6 takt ciklusa, pa zatim postavlja u procesor status registru masku za prekid u stanje logičke jedinice. Posle toga puni programski brojač vektorom (adresom) koju nalazi na lokacijama $FFFC i $FFFD. Na taj način se obavlja indirektan skok na reset rutinu. Kod Komodora ona počinje na adresi $FCE2.

$\overline{\text{IRQ}}$, zahtev za prekidom (engl. interrupt request), ulaz

Kada se na ovom ulazu pojavi logička nula, mikroprocesor završava trenutno započetu naredbu, a zatim prekida izvršavanje tekućeg programa. U tom trenutku on ispituje stanje maske prekida iz procesor status registra (engl. I flag, interrupt mask). Ako maska nije postavljena na logičku jedinicu, prekid je dozvoljen. Programski brojač i procesor status registar se stavljaju na stek. Procesor zatim postavlja masku u stanje logičke jedinice da bi onemogućio ponovnu pojavu prekida. Na kraju se programski brojač puni vektorom (adresom) sa adrese $FFFE (niži bajt) i $FFFF (viši bajt). Na taj način se vrši indirektan skok na program za obradu prekida. Kod Komodora on počinje na adresi $FF48.

$\overline{\text{NMI}}$, nemaskirajući prekid (engl. non maskable interrupt), ulaz

Ovo je ulaz koji se aktivira negativnom ivicom (prelazak sa logičke jedinice na logičku nulu). Ovaj zahtev za prekidom se uvek prihvata tj. ne može se maskirati kao prethodni. Po aktiviranju ovog signala, sadržaj programskog brojača i procesor status registra se stavlja na stek, a zatim se sa lokacija $FFFA i $FFFB uzima adresa i stavlja u programski brojač. Na taj način se skače indirektno na program za obradu prekida. Kod Komodora on počinje na adresi $FE43.

Ukoliko istovremeno dođe do više zahteva za prekidom, oni će se opslužiti po sledećem prioritetu:

tip prekida	prioritet	adresa vektora (hex)	
		viši bajt	niži bajt
RESET	1 (najviši)	FFFD	FFFC
$\overline{\text{NMI}}$	2	FFFB	FFFA
BRK	3	FFFF	FFFE
$\overline{\text{IRQ}}$	4 (najniži)	FFFF	FFFE

AEC, (engl. address enable control), kontrolni ulaz

Adresa na adresnoj magistrali je ispravna samo ako je ova linija na logičkoj jedinici. Ukoliko je na logičkoj nuli, adresne linije su u stanju visoke impedanse.

Na taj način se prepušta kontrola nad adresnom magistralom, nekoj drugoj jedinici.

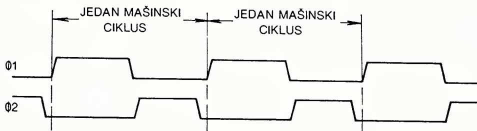
R/$\overline{\text{W}}$, (engl. read/$\overline{\text{write}}$), kontrolni izlaz

Ovaj signal generiše mikroprocesor da bi kontrolisao smer podataka na magistrali podataka. Ova linija je u stanju logičke jedinice uvek osim u slučaju kada mikroprocesor upisuje podatke u memoriju ili perifernu jedinicu.

Vcc, napajanje +5V
GND, masa

Vremenski dijagrami

Pomoću vremenskih dijagrama se prikazuju signali (naponi) na pojedinim izvodima mikroprocesora za vreme obavljanja pojedinih operacija. Referentni signal za sve događaje je takt jer se sve događa sinhrono sa njim. Mikroprocesor 6510 koristi relativno jednostavnu kombinaciju dva takt signala. To je tzv. dvofazni takt. Svaka perioda signala Φ1 i Φ2 čini jedan mašinski ciklus.

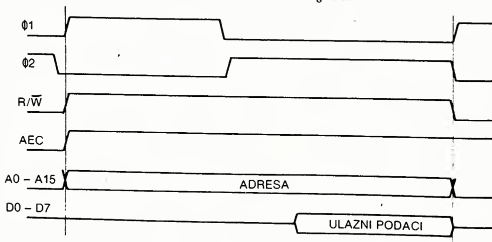


Sl. 11. 3. Dvofazni takt

Mikroprocesor izvršava program tako što iz memorije učita kôd naredbe, izvršava tu naredbu, učitava kôd sledeće naredbe, izvršava je i tako redom.

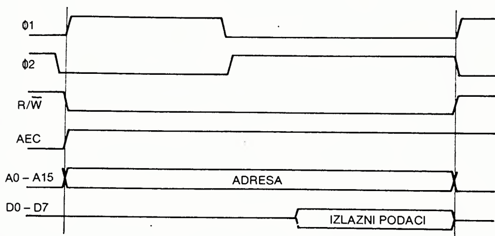
Postoje tri tipa mašinskih ciklusa:

1) operacija čitanja – kada se podatak sa magistrale učitava u CPU
2) operacija upisivanja – kada CPU stavlja na magistralu svoj podatak
3) interna operacija – kada nema aktivnosti na magistrali



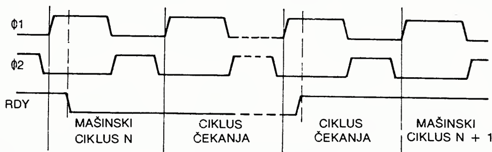
Sl. 11. 4. Standardni mašinski ciklus čitanja

Sve naredbe se sastoje od kombinacije ovih mašinskih ciklusa. Za pojedine naredbe je potrebno od dva do sedam mašinskih ciklusa.



Sl. 11. 5. Standardni mašinski ciklus upisivanja

Kod ubacivanja ciklusa čekanja linija RDY mora da pređe sa logičke jedinice na logičku nulu u toku aktivnog stanja takta Φ1 i to za vreme mašinskog ciklusa koji nije upisivanje.



Sl. 11. 6. Način ubacivanja ciklusa čekanja

Ukoliko se RDY signal aktivira za vreme mašinskog ciklusa upisivanja, ciklusi čekanja će se i dalje ubacivati ali će se pojaviti i u prvom sledećem ciklusu koji nije upisivanje.

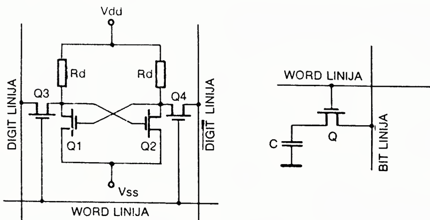
11.2 RAM

RAM je skraćenica od engleskih reči Random Access Memory, što znači da se radi o memoriji kod koje se direktno može prići bilo kojoj memorijskoj ćeliji. U jednu ćeliju se podatak može upisati ili iz nje pročitati.

RAM memorije su veoma brze, što znači da se podaci sa njima razmenjuju za veoma kratko vreme (nekoliko stotina nanosekundi). U slučaju isključivanja napona napajanja celokupan sadržaj memorije se gubi.

Postoje dve vrste RAM-a: statički i dinamički. Memorijske ćelije statičkog RAM-a se sastoje od nekoliko tranzistora. Oni čine kolo koje ima dva stabilna stanja (flip-flop). Jedno stanje odgovara logičkoj nuli, a drugo logičkoj jedinici. Upisivanjem podatka kolo se prebacuje u odgovarajuće stanje i u njemu ostaje do upisivanja sledećeg podatka ili do gubitka napajanja.

Memorijska ćelija dinamičkog RAM-a se sastoji od jednog MOS (metal oxide semiconductor) tranzistora koji podatak pamti samo vrlo kratko vreme. Da bi se podatak sačuvao za duži period, potrebno je obavljati osvežavanje memorijske ćelije (na primer svake druge milisekunde). Pri čitanju sadržaja jedne memorijske ćelije njen sadržaj se gubi i zbog toga je potrebno da se on odmah ponovo upiše. Na taj način se u toku procesa čitanja vrši istovremeno i osvežavanje. Dinamički RAM je komplikovaniji za upotrebu, ali je u slučaju većih memorijskih kapaciteta jeftiniji od statičkog jer se njegova memorijska ćelija sastoji od samo jednog tranzistora.

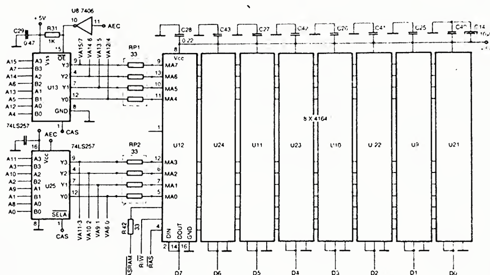


Sl. 11. 7. Ćelija statičke i dinamičke memorije

U Komodoru se koristi dinamički RAM. Postoji ukupno 8 memorijskih integrisanih kola (U9—U12 i U24—U24) tipa 4164. Svako kolo je kapaciteta 64 Kbita tj. sadrži 65536 ćelija veličine 1 bit. Svaka linija podataka ovih kola je vezana na odgovarajuću liniju magistrale podataka. Na taj način sva kola zajedno čine memoriju od 64 Kbajta. Ovo u potpunosti pokriva ceo adresni prostor mikroprocesora 6510, ali kao što se vidi iz organizacije memorije, ceo RAM nije uvek iskorišćen. Na nekim adresama koje pokrivaju RAM nalazi se ROM i registri ulazno izlaznih jedinica. U slučaju da se adresira neka od ovih dupliranih ili tripliranih lokacija, logika za dekodovanje će omogućiti aktiviranje samo jedne jedinice i to u zavisnosti od obavljene operacije.

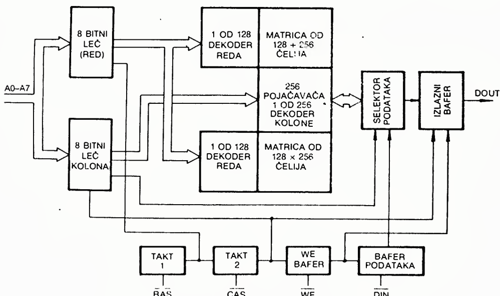
U Komodorovom RAM-u memorijske ćelije su organizovane u 256 redova i 256 kolona. Broj izvoda na integrisanom kolu je smanjen multipleksiranjem adrese, što znači da se u njega dovodi jedan deo adrese, a za njim drugi. U ovom slučaju osam adresnih linija memorijskog kola vezano je na izlaze multipleksera U13 i U25 (74LS257) čiji su ulazi vezani na adresnu magistralu.

Signale $\overline{CAS}$ (Column address strobe) i $\overline{RAS}$ (Row address strobe) generiše video kontroler (VIC II). Signal $\overline{CASRAM}$ se dovodi sa adresnog dekodera. U trenutku kada je aktivan signal $\overline{CAS}$, multiplekser propušta adrese sa linija A tj. A15 do A8. Kada $\overline{CAS}$ nije aktivan, propuštaju se adrese sa linija B tj. A7 do A0. Međutim, ako je AEC signal na logičkoj nuli ($\overline{AEC}$=1), izlazi multipleksera će biti u stanju visoke impedanse. Adresiranje memorije sada obavlja video kontroler preko svojih adresnih linija VA15/7 do VA8/0. Ove adrese su već multipleksirane u samom video kontroleru.



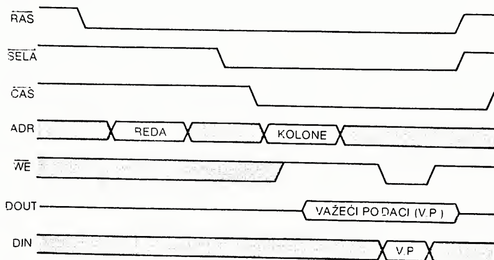
Sl. 11. 8. 64K RAM

Dinamička memorija će izgubiti podatke ako se svakom redu ne pristupi barem jedanput svake 2ms. O ovome vodi računa video kontroler koji se obraća memoriji nezavisno od procesora. On to radi za vreme faze 1 procesorskog takta (tj. kada je $\Phi1=1$, a $\Phi2=0$) kada procesor ne obavlja nikakve eksterne operacije.

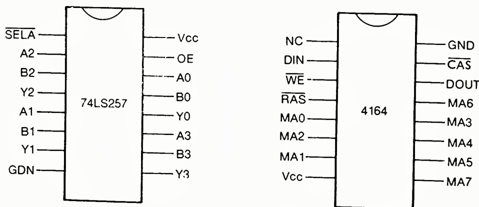


Sl. 11. 9. Unutrašnja organizacija dinamičkog RAM-a

Pri adresiranju memorije prvo se dovodi $\overline{RAS}$ signal. Zatim sledi adresa reda. Za njom se šalje $\overline{CAS}$ signal pa adresa kolone. Ukoliko je $\overline{WE}$ na logičkoj jedinici, podaci mogu biti pročitani na izlazu Dout. Ako je $\overline{WE}=0$, podaci sa Din mogu biti upisani u memoriju. Ako se signal $\overline{RAS}$ drži aktivnim, mogu se menjanjem adrese kolone adresirati sve lokacije u okviru datog reda.



Sl. 11. 10. Vremenski dijagrami pri čitanju i upisivanju u RAM



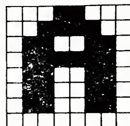
Sl. 11. 11. Raspored izvoda na kolima 74LS257 i 4164

11.3 **ROM**

U ROM memoriju (engl. read only memory) podaci se upisuju u toku procesa proizvodnje integrisanog kola i više se ne mogu menjati. Ne brišu se ni prilikom nestanka napona napajanja. Komodorov ROM čine 3 integrisana kola:

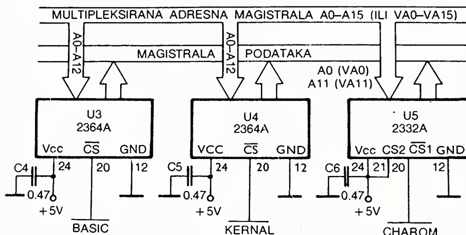
1. Bejzik ROM (U3) tipa 2364A kapaciteta 8 Kbajta. U ovom kolu je smešten Bejzik interpreter.

2. Kernal ROM (U4) tipa 2364A kapaciteta 8 Kbajta. U ovom kolu je smešten operativni sistem Komodora – Kernal.
3. Karakter ROM (U5) tipa 2332A kapaciteta 4 Kbajta. Ovde su smeštena oba skupa Komodorovih karaktera:
   – velika slova i grafički simboli 1 i grafički simboli 2
   – mala slova, velika slova i grafički simboli 2

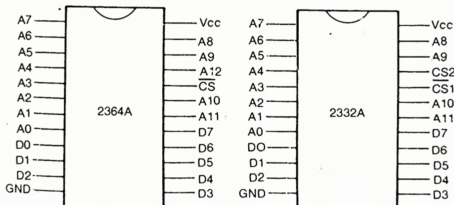


Sl. 11.12. Karakter slova A

Karakteri su organizovani u matrice od 8×8 tačaka. Za pamćenje jednog karaktera potrebno je 8 bajtova odnosno 64 bita, jer svakoj tački na ekranu odgovara jedan bit u memoriji. Ako je bit postavljen na jedinicu, na odgovarajućem mestu na ekranu će tačka postati vidljiva, a u suprotnom neće postojati.



Sl. 11. 13. 20K ROM



Slika 11. 14. Raspored izvoda na kolima 2364A i 2332A

S obzirom da je za svaki karakter potrebno 8 bajtova, a da postoje dva skupa od po 256 karaktera vidi se da je za sve karaktere potrebno 2×8×256=4 Kbajta memorije ROM-a. Pristup pojedinom karakteru se obavlja pomoću njegovog ekranskog koda. Ekranski kod se uzima iz ekranske memorije (obično od adresa $0400 do $0800) i dodaje na baznu adresu karakter ROM-a.

Pojedini ROM-ovi se aktiviraju preko $\overline{CS}$ linija na koje se dovode upravljački signali iz logike za upravljanje memorijom. Adresni prostori koje zauzimaju pojedini ROM-ovi su:

bejzik ROM	$A000 – $BFFF
Kernal ROM	$E000 – $FFFF
Karakter ROM	$D000 – $DFFF

11.4 VIDEO KONTROLER

Video kontroler je specijalizovano integrisano kolo koje upravlja prikazivanjem podataka na ekranu.

U Komodoru se koristi video kontroler tipa 6567 (VIC II). Glavnu ulogu u upravljanju sistemom ima upravo on jer obavlja operacije koje su vremenski kritične. Kontrolu nad sistemom VIC II preuzima u trenutku kada je aktivna faza 1 dvofaznog procesorskog takta ($\Phi 1=1$, $\Phi 2=0$).

Kod ovakvog načina deljenja magistrale, svi pristupi memoriji moraju da budu kompletirani u jednoj polovini mašinskog ciklusa tj. u intervalu nešto manjem od 500ns uključući postavljanje adrese, pristup podacima i prihvatanje podataka. Ponekad VIC II zahteva obraćanje memoriji u kraćim vremenskim intervalima od 500ns; na primer, pristup ekranskim kodovima u ekranskoj memoriji ili podacima o sprajtovima. Tada VIC II forsirano ubacuje procesoru cikluse čekanja preko RDY linije, a zatim koristi fazu 2 sistemskog takta ($\Phi 2=1$) za rad sa memorijom.

Funkcije pojedinih izvoda video kontrolera

DB0 do DB7, sistemska magistrala podataka, dvosmerne linije (ulazi ili izlazi)
Preko ovih osam izvoda razmenjuju se podaci između VIC II i memorije ili mikroprocesora. Ova magistrala je kontrolisana signalima $\overline{CS}$, R/$\overline{W}$ i $\Phi 0$. Za vreme dok su $\Phi 0$ i $\overline{AEC}$ na logičkoj jedinici i $\overline{CS}$ na logičkoj nuli, VIC II može pristupiti magistrali podataka.

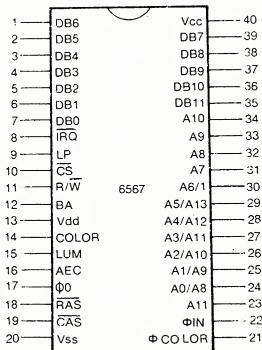
DB8 do DB11, lokalna magistrala podataka, ulazi
Preko ovih linija se učitavaju podaci o bojama iz kolor RAM-a.

A6/1, A5/A13 do A0/A8, multipleksirana adresna magistrala, dvosmerne linije
Ulazne – kada mikroprocesor adresira interne registre video kontrolera (A5–A0). Izlazne – kada kontroler adresira memoriju. U ovom slučaju adresne linije su multipleksirane i u kombinaciji sa kontrolnim linijama $\overline{CAS}$, $\overline{RAS}$ i A6 služe za adresiranje dinamičkog RAM-a. Važeće adrese su A6 do A0 za vreme aktivnog signala $\overline{RAS}$ dok su za vreme aktivnog signala $\overline{CAS}$ važeće adrese A11 do A7 i A6=1.

A11–A8, procesorska adresna magistrala, ulazi
Koristi se za adresiranje 2K×8 bajta karakter ROM-a. Adrese A7 do A0 moraju se izdvojiti iz multipleksirane adrese i privremeno zadržati na magistrali za vreme aktivnog $\overline{RAS}$ signala.

$\overline{CS}$, (engl. chip select), ulaz
Logička nula na ovom izvodu omogućuje pristup registrima kontrolera ali samo kada su AEC i Φ2 na logičkoj jedinici.

R/$\overline{W}$, (engl. read/$\overline{write}$), ulaz
Ovaj signal generiše mikroprocesor da bi kontrolisao smer podataka pri pristupu registrima kontrolera. Logička jedinica označava čitanje iz odgovarajućeg registra, dok logička nula označava upisivanje u registar. Ovo sve važi samo ako je $\overline{CS}$ na logičkoj nuli. U suprotnom se R/$\overline{W}$ linija ignoriše.



Sl. 11. 15. Raspored izvoda videokontrolera 6567

Φ0, sistemski takt, izlaz
Ovaj signal se vodi na sinhronizaciju u kolo za generisanje takta kao i na Φ1 ulaz mikroprocesora.

AEC, (engl. address enable control), izlaz
AEC linija omogućuje da se izlazi A15 do A0 na mikroprocesoru dovedu u stanje visoke impedanse. Ovaj signal je aktivan kada je na logičkoj nuli tako da omogućuje direktnu vezu s mikroprocesorom. Na taj način video kontroler preuzima kontrolu nad adresnom magistralom.

BA, (engl. bus available), izlaz
U normalnom stanju na ovoj liniji je logička jedinica. Međutim, ukoliko je potrebno da video kontroler ima pristup podacima i za vreme faze 2, kada čita ekranske kodove

ili podatke o sprajtovima, ova linija se postavlja na logičku nulu i to u fazi 1. Posle ovoga, procesor mora završiti obraćanje memoriji u sledeće tri faze 2. U četvrtoj fazi 2, AEC signal se postavlja na logičku nulu i video kontroler preuzima podatke.
Ekranski kodovi se čitaju svake osme linije i zahtevaju 40 uzastopnih pristupa ekranskoj memoriji (video matrici) u fazi 2.
Pokazivači podataka o sprajtovima se čitaju iz ekranske memorije svake faze 1 na kraju svake linije. Ukoliko je potrebno, koriste se i dodatni uzastopni ciklusi.

faza	podaci	čitanje
1	sprajt. pokazivač	u svakoj liniji
2	sprajt bajt 1	u svakoj liniji ako se sprajt prikazuje
1	sprajt bajt 2	– II –
2	sprajt bajt 3	– II –

$\overline{RAS}$, (engl. row address strobe), izlaz
$\overline{CAS}$, (engl. column address strobe), izlaz
Ove dve linije se koriste kao kontrolne pri upravljanju multipleksiranom adresnom magistralom. Oba signala se generišu svake faze 2, kao i pri svakom obraćanju memoriji od strane video kontrolera (u toku faze 1). Na taj način se osvežava dinamička RAM memorija nezavisno od mikroprocesora.

AEC	Φ0	$\overline{CS}$	R/$\overline{W}$	aktivnost VIC-a na magistrali
0	1	×	×	faza 1, čitanje podatka, osvežavanje
1	0	×	×	faza 2, čitanje podatka, procesor je otkačen
1	1	0	0	upisivanje u adresirani registar
1	1	0	1	čitanje iz adresiranog registra
1	1	1	×	bez aktivnosti

$\overline{IRQ}$, (engl. interrupt request), izlaz
$\overline{IRQ}$ linija se postavlja na logičku nulu ukoliko nastupi dozvoljeni prekid unutar VIC II. Postoje četiri moguća uzroka prekida:
1. IRST – kada je sadržaj brojača linija jednak sadržaju registra linija
2. IMDC – kada dođe do sudara sprajta sa podacima (samo prvi sudar)
3. IMMC – kada dođe do sudara dva sprajta (samo prvi sudar)
4. ILP – kada se pojavi negativna ivica na LP ulazu (jedanput u okviru poluslike)
Naravno, ovi prekidi mogu biti maskirani postavljanjem odgovarajuće maske na adresi SD01A. Pri tome za dozvolu prekida treba upisati u odgovarajući bit logičku jedinicu.

adresa:	7	6	5	4	3	2	1	0	
D019	IRQ	–	–	–	ILP	IMMC	IMMBC	IRST	registar prekida
D01A	–	–	–	–	ELP	EMMC	EMBC	ERST	registar maske

IRQ bit u registru prekida je invertovan u odnosu na signal na izvodu $\overline{IRQ}$.

$\overline{LP}$, (engl. light pen), ulaz
Na ovu liniju se priključuje svetlosna olovka ili prekidač paljbe džojstika 1. Okidanje se vrši negativnom ivicom, kada i dolazi do generisanja ILP prekida.

Φ in, ulaz

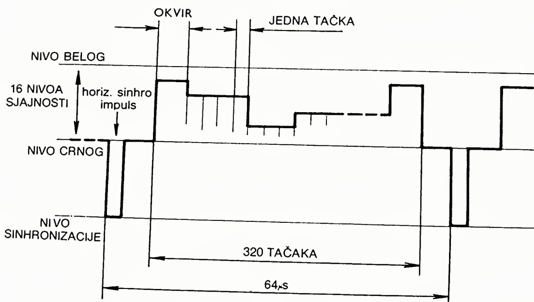
Na njega se dovodi signal DOT CLOCK sa generatora takta. On je za PAL sistem približno 7.88 MHz. To je frekvencija generisanja tačaka uračunavajući i one koje se ne vide za vreme povratnih intervala mlaza. Ova frekvencija se u VIC II deli sa 8 da bi se dobio takt Φ0.

Φ col, ulaz

Na ovaj izvod se dovodi signal frekvencije 17.734472 MHz. Njegovim deljenjem sa 4 dobija se frekvencija nosećeg signala boje tj. 4433618.75 Hz.

SINC + LUM, izlaz

Na ovom izvodu se dobija takozvani kompozitni luminentni signal. To je signal koji nosi informacije o sjajnosti pojedinih tačaka kao i sve potrebne impulse za sinhronizaciju. S obzirom da kod Komodora postoji 16 boja, luminentni signal može da dâ jednu od 16 gradacija sive boje (uključujući crnu i belu). Ovaj izlaz je sa otvorenim drejnom pa zahteva otpornik prema Vcc od oko 500 oma.

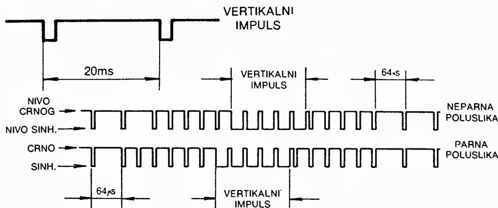


Sl. 11. 16. Izgled kompozitnog signala u intervalu jedne linije

Analiziranjem vertikalnih i horizontalnih sinhro impulsa zaključujemo da video kontroler VIC II daje signal za sistem sa poluslikom (engl. interlaced). Naime, svaka slika se sastoji iz dve poluslike: parne i neparne. Slike se smenjuju frekvencijom od 25 Hz, a poluslike frekvencijom od 50 Hz. U toku parne poluslike generišu se linije 2, 4, 6..., a zatim se mlaz vraća pa se u toku neparne poluslike generišu linije 1, 3, 5....

Zbog toga je perioda vertikalnih sinhro impulsa 20 ms. Unutar vertikalnih impulsa, nalaze se horizontalni impulsi da bi se oscilator horizontalnih sinhro impulsa održao u sinhronizaciji za vreme vertikalnog povratka mlaza.

Iz razloga nesimetričnosti parne i neparne poluslike, pre i posle implusa za vertikalnu sinhronizaciju ubacuju se, takozvani, izjednačavajući impulsi.



Sl. 11. 17. Impulsi za sinhronizaciju u parnoj i neparnoj poluslici

COLOR, izlaz

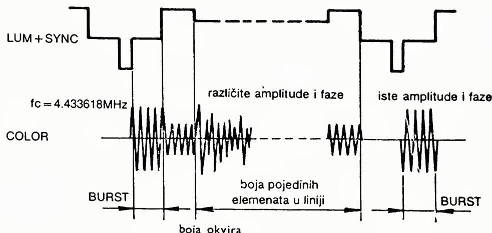
Na izlazu COLOR se dobija takozvani hrominentni signal. On svojom amplitudom nosi podatak o zasićenju, a svojom trenutnom fazom podatak o dominantnoj talasnoj dužini boje.

Potrebni podaci o učestanosti za sinhronizaciju generatora referentnog signala u sinhronom detektoru TV ili monitoru, generišu se na zadnjem stepeniku horizontalnog sinhro impulsa. Sinhronizacioni signal boje sačinjava grupa sinusoida nazvanih „color burst" (engl.).

COLOR izvod je sa otvorenim sorsom i zahteva otpornik prema masi od oko 1000 oma.

Vdd, ulaz — Napajanje +12 V

Vcc, ulaz — Napajanje +5 V can



Sl. 11. 18. Položaj sinhronizacionog signala boje u odnosu na sinhro impulse

Povezivonje video kontrolero so mikroprocesorom

Veza video kontrolera sa mikroprocesorom se ostvaruje na dva načina:

1. Procesor adresira pojedine registre video kontrolera u cilju njihovog čitanja ili upisivanja. Registri se nalaze u ulazno/izlaznom adresnom prostoru mikroprocesora.

2. Video kontroler preuzima kontrolu nad magistralama u cilju adresiranja memorije. Pomoću signala AEC postavlja adresne linije procesora u stanje visoke impedanse a pomoću BA signala ubacuje cikluse čekanja preko RDY ulaza procesora.

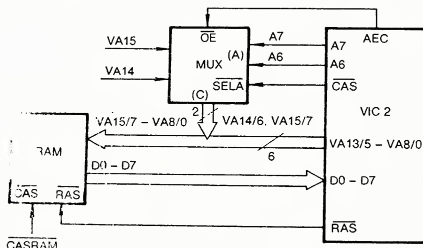
Povezivonje video kontrolero so memorijom

v e z a s a R A M-o m

VIC II je konstruisan da radi sa dinamičkim RAM-om. Zbog toga on obezbeđuje multipleksirane adrese kao i potrebne funkcije za osvežavanje, nezavisno od procesora. Multipleksiranjem adresa A7 i A15 sa VA15 i VA14 (koje daje CIA 2), moguće je adresirati ceo memorijski prostor u segmentima od po 16 KB.

Postoje dva načina čitanja (VIC II ne može da upisuje u RAM):

1. karakter način
2. bit mapirani način rada



Sl. 11. 19. Veza video kontrolera sa RAM-om

U prvom slučaju adresira se oblast od 1000 bajtova (25 linija × 40 karaktera) u memoriji. Ova oblast se zove video matrica. Lokacija video matrice je određena sadržajem registra na adresi SD018. U video matrici se nalaze karakteri prikazani preko svojih ekranskih (ne ASCII) kodova.

SD018	VM13	VM12	VM11	VM10	CB13	CB12	CB11	–

A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
VM13	VM12	VM11	VM10	VC9	VC8	VC7	VC6	VC5	VC4	VC3	VC2	VC1	VC0

Formiranje adrese karaktera u video matrici

VM13 do VM10 određuju početnu adresu video matrice, dok VC9 do VC0 određuju adresu pojedinih karaktera u okviru video matrice. Vidi se da video matrica može zauzeti 16 lokacija u okviru jednog segmenta od 16 KB. Pošto ovih segmenata ima 4, to je ukupno 64 lokacija. Po uključenju računara, video matrica je na adresi $0400.

Pošto je na osnovu prethodno definisane adrese adresiran određen karakter u okviru video matrice, čita se njegov ekranski kôd (8 bita) i on se dodaje na takozvanu bazu karaktera. Na taj način se dobija adresa gde je smešten karakter u obliku 8 × 8 matrice.

A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
CB13	CB12	CB11	D7	D6	D5	D4	D3	D2	D1	D0	RC2	RC1	RC0

CB13, CB12 i CB11 se uzimaju iz registra na adresi $D018. To je početna adresa baze karaktera.

D7 do D0 čine ekranski kôd karaktera.

RC2, RC1 i RC0 su brojač linija u okviru jednog karaktera.

Video kontroler čita 40 uzastopnih lokacija iz video matrice. Za to vreme procesor je u ciklusu čekanja.

U bit mapiranom režimu rada, kontroler čita bit po bit iz memorije i oni odgovaraju pojedinim tačkama na ekranu. Evo kako se formira adresa pojedinačne tačke:

A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
CB13	VC9	VC8	VC7	VC6	VC5	VC4	VC3	VC2	VC1	VC0	RC2	RC1	RC0

Bit CB13 je iz registra na adresi $D018. Pomoću njega se bira prva ili druga polovina od 8KB u okviru jednog segmenta od 16KB.

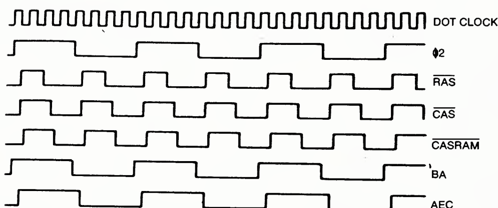
VC9 do VC0 čine brojač isti kao kod video matrice — broj pozicija (40 × 25 maksimum).

RC2 do RC0 čine brojač linija.

Na taj način se bajtovi adresiraju tako da svakih 8 susednih bajtova čine matricu 8 × 8 bita.

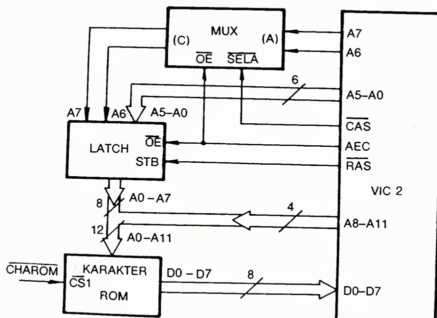
	bajt 0	8	16		312
	bajt 1	9	17		313
	bajt 2	10	18		314
	bajt 3	11	19		315
0. red	bajt 4	12	20		316
	bajt 5	13	21		317
	bajt 6	14	22		318
	bajt 7	15	23		319

	320	328	. . . .
	321	"	
	322	"	
	323	"	
1. red	324	"	
	325	"	
	426	"	
	327	"	



Sl. 11. 20. Vremenski dijagrami napona na kontrolnim linijama video kontrolera

Treba napomenuti da se zbog rada sa poluslikama i ovde prvo čitaju neparni bajtovi, a zatim parni, što daje celu sliku.



Sl. 11. 21. Veza video kontrolera sa karakter ROM-om

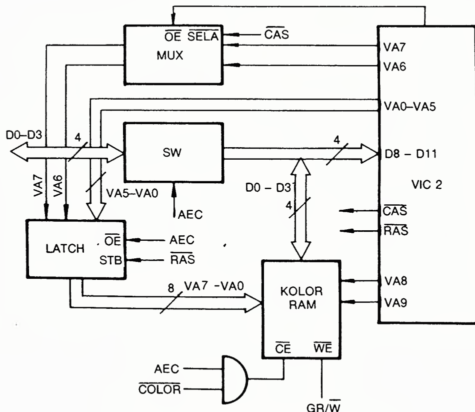
veza sa karakter ROM-om

Kada se linija $\overline{CHAROM}$ postavi na logičku nulu i baza karaktera pokazuje na $D000, mogu se čitati karakteri iz ROM-a. Ovaj ROM je kapaciteta 2KB. Pošto je za svaki karakter potrebno 8 bajtova, to je ukupno 512 karaktera. To su dva seta od po 128 karaktera, kao i isto toliko inverznih karaktera.

Adresiranje ROM-a se obavlja direktno adresnim linijama A11 do A0.

veza sa kolor RAM-om

Kolor RAM je postavljen fiksno u adresnom prostoru i zauzima lokacije od $D800 do $DBFF. On je kapaciteta 1K × 4 bita. Kada je video kontroler u karakter načinu rada, 1000

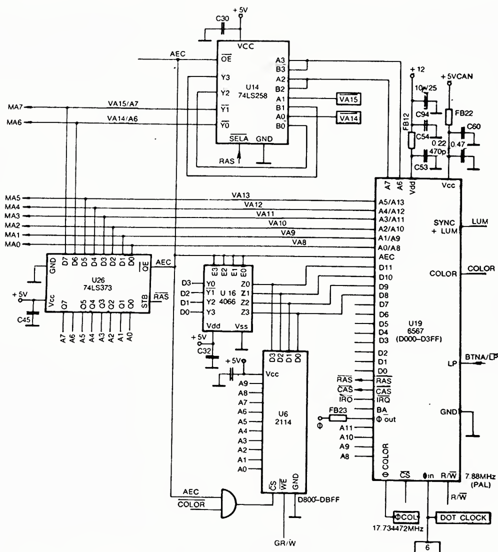


Sl. 11. 22. Veza vide kontrolera sa kolor RAM-om

lokacija u ovoj memoriji su tretirane kao 40 × 25 tj. svakom karakteru je pridruženo još 4 bita (jedna od 16 boja).

U bit mapiranom načinu rada kolor RAM se ne koristi, već njegovu funkciju preuzima video matrica. Svaki njen bajt je podeljen na 2 nibla (1 nibl je pola bajta) po 4 bita koji određuju dve boje (iz palete od 16 boja) koje mogu da budu prikazane u matrici 8 × 8.

Kolor RAM se adresira direktno linijama VA9 do VA0.

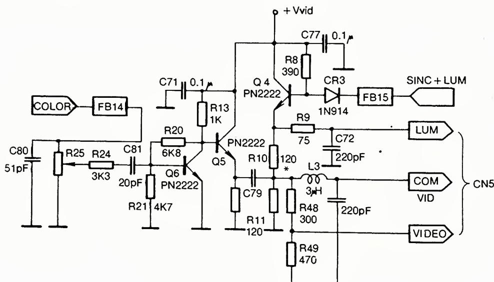


Sl. 11. 23. Detaljna šema veza video kontrolera sa sistemom

Video izlazni stepen

Luminentni i hrominentni (kolor) signal sa izlaza video kontrolera se nezavisno pojačavaju.

Luminentni signal se dovodi na tranzistor Q4 preko diode CR3. Q4 radi kao pojačavač sa zajedničkim kolektorom (emiter folover). Na taj način se obezbeđuje vrlo niska izlazna impedansa koja se veštački podiže na standardnu vrednost od 75 oma otpornikom R9. Ovo se radi zbog izbegavanja refleksija na video kablovima koji imaju karakterističnu impedansu od 75 oma.



Sl. 11. 24. Video izlazni stepen

Hrominentni signal se preko trimer potenciometra R 25, kojim se reguliše njegov nivo, vodi na tranzistor Q6 koji ga pojačava, a zatim na emiter folover Q5. Sa njegovog emitera, signal se preko razdvojnog kondenzatora C79 vodi na tačku sabiranja (obeleženu sa *) sa luminentnim signalom. Na taj način se dobija kompozitni kolor video signal. Preko razdelnika R 48 i R 49 on se kao video signal vodi na RF modulator.

11.5 AUDIO KONTROLER

U Komodoru se koristi specijalizovano kolo za generisanje zvuka tipa 6581 nazvanog SID (engl. sound interface device).

Funkcije pojedinih izvoda audio kontrolera

CAP1A, CAP1B, CAP2A, CAP2B, priključci za kondenzatore C1 i C2

Ovi kondenzatori određuju graničnu frekvenciju programabilnog filtera. Filter je drugog reda tipa „state variable" – (engl.) sa nezavisno promenljivim Q faktorom i graničnom frekvencijom. Oba parametra mogu se menjati softverski. U Komodoru su kondenzatori C1 i C2 vrednosti od po 2200pF pa je granična frekvencija 12 KHz. Inače opšti oblik izraza za graničnu frekvenciju je

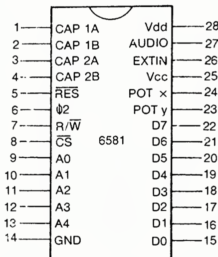
$$fcmax = \frac{2.6}{C} \times 10^{-5} \, Hz$$

Opseg filtra je 9 oktava.

$\overline{RES}$, ulaz

Ukoliko je za vreme od najmanje 10 uzastopnih Φ2 ciklusa na ovom izvodu prisutna logička nula, svi unutrašnji registri SID-a se postavljaju na nulu, a audio izlaz se potpuno utišava. Ovaj izvod je normalno priključen na procesorsku RESET liniju.

Sl. 11. 25. Raspored izvoda na kolu 6581



Φ2, ulaz, sistemski takt od 0.98 MHz

R/$\overline{W}$, ulaz

Ovaj signal generiše mikroprocesor. Na ovom izvodu postavlja logičku nulu ili jedinicu, u zavisnosti od toga da li čita ili upisuje podatke u registre SID-a.

$\overline{CS}$, ulaz

Logička nula na ovom izvodu omogućuje pristup registrima audio kontrolera, ali samo kada je Φ2 na logičkoj jedinici.

A0 do A4, ulazi

Adresne linije koje se koriste za adresiranje pojedinih registara u SID-u.

GND, masa

D0 do D7, magistrala podataka, dvosmerne linije

Preko ovih linija procesor upisuje ili čita određene podatke iz registara SID-a.

POT Y, ulaz A/D konvertora

A/D konvertor je integratorskog tipa, što znači da meri vreme za koje se preko otpornika R napuni kondenzator C na određenu vrednost napona. Za punu skalu vrednost ove RC konstante je određena sa

$$RC = 4.7 \times 10^{-4} \quad \text{sec}$$

POT X, ulaz, isto važi kao i za ulaz POT Y

VCC, napajanje od +5V

EXTIN, ulaz

Ovaj analogni ulaz omogućuje priključivanje spoljašnjeg audio signala i njegovo mešanje sa signalom iz SID-a, ili njegovu obradu kroz filter. Ulazna impedansa ovog priključka je reda 100 Koma, a izvod je na potencijalu od 6V. Maksimalna dozvoljena amplituda ulaznog napona je 3 Vpp. Ukoliko se ne vrši obrada filterom, od ovog ulaza do audio izlaza postoji jedinično pojačanje. Zbog toga se EXTIN može koristiti za lančano povezivanje (engl. daisy chain) više audio kontrolera. Softverski programabilna finalna jačina zvuka na izlazu ima uticaja i na signal koji dolazi preko EXTIN.

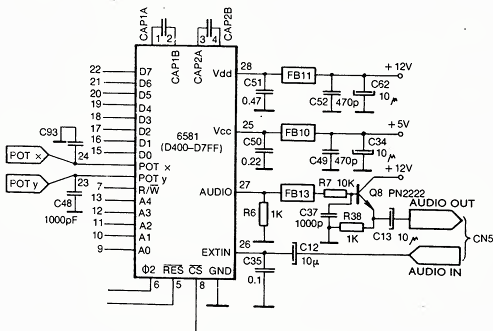
AUDIO, izlaz

To je izlaz izlaznog stepena sa otvorenim sorsom koji je spojen otpornikom od oko 1 Kom na masu. Izlazna amplituda dostiže maksimum 2 Vpp sa jednosmernim nivoom od + 6V. Zbog toga se za spregu preporučuje upotreba razdvojnog kondenzatora.

Vdd, napajanje +12V

Veza sa sistemom

Izlazni audio signal se preko emiter folovera QB i razdvojnog kondenzatora C13 vodi na konektor, kao i na RF modulator.



Sl. 11. 26. Veza audio kontrolera sa sistemom

11.6 PERIFERNE JEDINICE I PRIKLJUČCI

11.6.1 RF modulator

U RF modulatoru se kompozitni video kolor signal sa audio signalom koristi za modulaciju nosioca frekvencije od 590MHz (TV kanal 36 UHF). Ova frekvencija se može podesiti pomoću promenljivog kondenzatora „TRIM" sa zadnje strane računara. Na taj način se može fino podesiti kanal, ukoliko televizor nema takvu mogućnost.

Sl. 11. 27. Povezivanje RF modulatora

11.6.2 **Priključak za kasetofon**

Za rad sa kasetofonom koriste se sledeće linije:

CASS RD, linija za čitanje podataka sa kasetofona. Dovodi se na $\overline{FLAG}$ ulaz integrisanog kola U1 (CIA1).

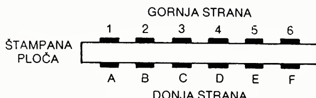
CASS WR, linija za upisivanje podataka na kasetofon. Povezana je sa izlazom P3 mikroprocesora 6510.

CASS SENSE, linija za detekciju pritisnutog „PLAY" tastera na kasetofonu. Dovodi se na ulaz P4 mikroprocesora 6510.

Sl. 11. 28. Priključak za kasetofon

CASS MOTOR, komandna linija za pogon motora kasetofona. Povezana je sa izlazom P5 mikroprocesora 6510. Kada je priključak P5 na logičkoj jedinici, tranzistor Q2 ide u zasićenje, pa je baza tranzistora Q1 približno na naponu 0V, te je on zakočen. Motor, prema tome, nema napajanja. Kada je P5 na logičkoj nuli, Q2 je zakočen pa struja postoji kroz otpornik R2 i zener diodu CR2. Ovo obezbeđuje napon od oko 7.5V na bazi Q1. Q1 i Q3 provode, a na emiteru Q3 je napon od 7.5 – 2*Vbe = 6V jer je napon Vbe = 0.75V. Na taj način je napajanje motora stabilisano, odnosno njegova brzina okretanja je kontrolisana.

Tranzistori Q1 i Q3 su u Darlington sprezi da bi se obezbedilo veliko strujno pojačanje, jer je za pokretanje motora potrebna veća struja.

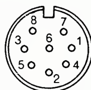


Sl. 11. 29. Izgled priključka CN3 za priključivanje kasetofona. Pogled za zadnje strane računara

pin	funkcija
A,1	GND
B,2	+5V
C,3	CASS MOTOR
D,4	CASS RD
E,5	CASS WR
F,6	CASS SENSE

11.6.3 **Audio – video priključak**

pin	funkcija
1	luminentni signal
2	GND
3	audio izlaz
4	kompozitni video signal
5	audio ulaz
6	NC
7	NC ne koriste se
8	NC



Sl. 11. 30. Izgled AUDIO-VIDEO priključka

U nekim varijantama se koriste 5 – pinski DIN konektori. Raspored izvoda je isti kao i kod 8 – pinskih.

11.6.4 **Kompleksni interfejs adapter (CIA)**

Kompleksni interfejs adapter (CIA) je specijalizovano integrisano kolo preko koga se obavlja komunikacija između računara i spoljašnjeg sveta. U Komodoru postoje dva ovakva kola i njima su dodeljene različite funkcije. Svaka CIA se sastoji od 16 registara koji su prikazani u sledećoj tabeli.

registar br.	lokalna adresa	ime	funkcija
1	0	PRA	registar podataka A
2	1	PRB	registar podataka B
3	2	DDRA	registar smera podataka za reg. A
4	3	DDRB	registar smera podataka za reg. B
5	4	TALO	tajmer A donji bajt
6	5	TAHI	tajmer A gornji bajt
7	6	TBLO	tajmer B donji bajt
8	7	TBHI	tajmer B gornji bajt
9	8	TOD 10THS	registar desetinki sekunde
10	9	TOD SEC	registar sekundi
11	A	TOD MIN	registar minuta
12	B	TOD HR	registar sati
13	C	SDR	registar serijskih podataka
14	D	ICR	registar za kontrolu prekida
15	E	CRA	kontrolni registar A
16	F	CRB	kontrolni registar B

Registri 1 i 3 čine kapiju A, dok registri 2 i 4 čine kapiju B.

PRA i PRB mogu biti ulazni ili izlazni, ili uzlazno-izlazni, u zavisnosti od sadržaja registara DDRA i DDRB. Ukoliko je neki bit u registru smera podataka postavljen na logičku nulu, odgovarajući bit u registru podatka će biti izlazni. U slučaju logičke jedinice, odgovarajući bit će biti ulazni.

Primer:

DDRA	PRA	smer toka podataka
1	PA7 <--	ulazni
0	PA6 -->	izlazni
1	PA5 <--	ulazni
1	PA4 <--	ulazni
0	PA3 -->	izlazni
0	PA2 -->	izlazni
1	PA1 <--	ulazni
0	PA0 -->	izlazni

Isto važi i za kapiju B.

CIA sadrži u sebi časovnik realnog vremena, kao i dva šesnaestobitna interval tajmera koji se mogu programirati ili čitati, u zavisnosti od sadržaja kontrolnih registara A i B.

Svaki interval tajmer se sastoji od šesnaestobitnog brojačkog registra, iz koga podaci mogu samo da se čitaju, i šesnaestobitnog tajmera tipa leč (engl. latch) u koji podaci mogu da se upisuju. Tajmeri A i B se mogu povezati ukoliko su potrebni duži vremenski intervali.

Kontrolni registar A (CRA)

CRA je pridružen tajmeru A i njegovi pojedini biti imaju sledeću funkciju:

bit ime	stanje
0 START	1 – startuje tajmer A 0 – zaustavlja tajmer A
1 PBON	1 – signal završenog brojenja tajmera A se pojavljuje na izvodu PB6 kapije B. Ova funkcija ima prioritet nad onom definisanom u DDRB i ona forsirano postavlja PB6 u funkciju izlaza. Ukoliko je PBON na logičkoj nuli, PB6 je postavljen na osnovu DDRB.
2 OUTMODE	1 – Po završetku brojanja tajmera A na izvodu PB6 će se promeniti nivo sa logičke nule na logičku jedinicu. Povratak na nulu se vrši resetovanjem cele CIA. 0 – Po završetku brojanja tajmera A na izvodu PB6 će se pojaviti impuls trajanja jednog ciklusa.
3 RUNMODE	1 – monostabilan rad. Tajmer odbrojava od programirane vrednosti unazad do nule, generiše zahtev za prekidom, ponovo uzima vrednost iz leč registra i zatim se zaustavi. 0 – kontinuiran rad. Tajmer odbrojava od programirane vrednosti unazad do nule, generiše zahtev za prekidom, ponovo uzima vrednost iz leč registra, zatim ponovo odbrojava unazad do nule i tako beskonačno ponavlja ciklus.
4 LOAD	1 – Forsirano punjenje tajmer brojačkog registra sadržajem tajmer leč registra bez obzira da li brojač u tom trenutku broji ili ne. Ovo je impulsni signal (engl. strobe) tj. automatski se vraća na nulu, tako da upisivanje nule nema smisla. Prema tome, tajmer brojač se puni sadržajem tajmer leča posle odbrojavanja brojača do nule ili forsiranim punjenjem pomoću LOAD bita ili upisivanjem u gornji bajt tajmer leča ukoliko tajmer brojač stoji. Ukoliko brojač broji, upisivanje u gornji bajt će napuniti tajmer leč, ali neće imati efekta na brojač.
5 INMODE	1 – Tajmer odbrojava pozitivne ivice na izvodu CNT. 0 – Tajmer broji Φ2 impulse.
6 SPMODE	1 – Serijski izvod je izlaz. CNT izvod daje potreban takt za pomeranje (engl. shift). 0 – Serijski izvod je ulaz. Potreban je spoljašnji pomerački takt.
7 TODIN	1 – Sinhronizacija časovnika realnog vremena preko izvoda TOD frekvencijom od 50Hz (ovo važi za našu električnu mrežu). 0 – Važi isto za električnu mrežu od 60Hz.

Kontrolni registar B (CRB)

bit ime	stanje
0 START	Važi isto kao kod CRA samo za tajmer B.
1 PBON	1 – Signal završenog brojanja tajmera B se pojavljuje na izvodu PB7 kapije B. Ova funkcija ima prioritet nad onom definisanom u DDRB i ona forsirano postavlja PB7 kao izlaz. 0 – PB7 je postavljen na osnovu DDRB.

2 OUTMODE Važi isto kao kod CRA samo za tajmer B.
3 RUNMODE Važi isto kao kod CRA samo za tajmer B.
4 LOAD Važi isto kao kod CRA samo za tajmer B.
5,6 INMODE Kombinacija ova dva bita određuje jedan od četiri moguća ulazna signala za tajmer B:

bit 5	bit 6	
0	0	Tajmer B odbrojava $\Phi 2$ impulse.
0	1	Tajmer B odbrojava pozitivne ivice na izvodu CNT.
1	0	Tajmer B odbrojava signalne impulse završenog brojanja tajmera A.
1	1	Tajmer B odbrojava signalne impulse završenog brojanja tajmera A samo za vreme dok se CNT izvod drži u stanju logičke jedinice.

7 ALARM 1 – Upisivanje u registre časovnika realnog vremena postavlja vreme ALARM-a.
0 – Upisivanje u registre časovnika realnog vremena postavlja realno vreme.

lokalna adresa registra	ime	TODIN	SPMODE	INMODE	LOAD	RUNMODE	OUTMODE	PBON	START
SE	CRA	0 = 60Hz 1 = 50Hz	0 = IN 1 = OUT	0 = $\Phi 2$ 1 = CNT	1 = FORCE LOAD (strobe)	0 = cont. 1 = mon.	0 = impuls 1 = nivo	0 = PB60FF 1 = PB60N	0 = STOP 1 = START

TA (LOAD – START)

lokalna adresa registra	ime	ALARM	IN	MODE	LOAD	TUNMODE	OUTMODE	PBON	START
SF	CRB	0 = TOD 1 = ALARM	0 0 0 1 1 0 1 1	= $\Phi 2$ = CNT = TA = CNT TA	1 = FORCE LOAD (strobe)	0 = cont. 1 = nom.	0 = impuls 1 = nivo	0 = PB70FF 1 = PB70N	0 = STOP 1 = START

TB (IN – START)

Svi neiskorišćeni biti registra pri čitanju daju logičku nulu.

Registar serijskih podataka (SDR)

Serijski podaci se prenose preko osmobitnog sinhronog pomeračkog registarskog sistema. Kada je programiran da radi kao ulazni, podaci koji pristižu na SP izvod se pomeraju u pomeračkom registru pri svakoj pozitivnoj ivici signala dovedenog na izvod CNT. Posle 8 impulsa, podaci iz pomeračkog registra se prebacuju u registar serijskih podataka (SDR), a zatim se generiše zahtev za prekidom. Procesor sada može da pročita sadržaj SDR.

Kada serijski sistem radi kao izlazni, tajmer A se koristi kao bod rejt generator (broj bita u sekundi — engl. baud rate). Podaci se iz SDR prebacuju u pomerački registar, a zatim se iz njega pomeraju preko izvoda SP frekvencijom koja odgovara polovini intervala tajmera A. Prema tome, maksimalni bod rejt je Φ2/4, ali je maksimalna praktična vrednost određena opterećenjem linije kao i brzinom prijema podataka.

Takt signal dobijen iz tajmera A, pojavljuje se na izlazu CNT kao izlaz. Podaci postaju važeći u trenutku negativne ivice izlaznog takta na CNT izvodu i ostaju takvi sve do sledeće negativne ivice. Posle 8 impulsa generiše se zahtev za prekidom označavajući da je pomerački registar spreman za punjenje novim bajtom. Ako je SDR napunjen novim podatkom pre generisanja prekida, podatak se automatski prebacuje u pomerački registar, a zatim šalje na izlaz. Ukoliko nema više podataka za prenose, posle osmog CNT impulsa CNT ostaje na logičkoj jedinici, a SP ostaje na onom logičkom nivou koji odgovara poslednjem prenetom bitu.

Serijski podaci se šalju tako što ide prvo MSB (bit najveće težine), pa prema tome i ulazni serijski podaci moraju da budu u tom formatu.

Dvosmerne osobine izvoda SP i CNT omogućuju povezivanje više CIA jedinica na zajedničku komunikacionu magistralu na kojoj jedna radi tako što šalje podatke (engl. master), dok su ostale podređene tj. primaju podatke (engl. slave). SP i CNT izvodi su sa otvorenim drejnom što omogućuje direktno povezivanje.

Protokol za master/slave komunikaciju može se slati direktno preko same komunikacione magistrale ili preko posebnih linija.

Registri časovnika realnog vremena (TOD)

Časovnik realnog vremena se sastoji od četiri registra:

- registar desetinki sekunde (10THS)
- registar sekundi (SEC)
- registar minuta (MIN)
- registar sati (HR)

lokalna adresa	ime								
8	TOD 10THS	0	0	0	0	T8	T4	T2	T1
9	TOD SEC	0	SH4	SH2	SH1	SL8	SL4	SL4	SL1
A	TOD MIN	0	MH4	MH2	MH1	ML8	ML4	ML2	ML1
B	TOD HR	PM	0 0	0	HH	HL4	HL2	HL1	HL1

Svi podaci se direktno čitaju u BCD formatu radi lakšeg povezivanja na cifarske indikatore i slično. Časovi se računaju po sistemu AM/PM tj. može da bude maksimum 12 časova pre (PM = 0) i 12 časova po podne (PM = 1).

Časovnik se automatski zaustavlja pri bilo kakvom upisivanju u registar sati i ne nastavlja rad sve do upisivanja u registar desetinki sekunde. Ovo osigurava startovanje časovnika u precizno određenom željenom trenutku.

Zbog mogućnosti prenosa iz nižeg registra u viši u bilo kom trenutku, obezbeđena je leč funkcija radi čuvanja informacija u toku čitanja. Svi registri se lečuju u trenutku čitanja registra sati i ostaju u tom stanju sve do čitanja registra desetinki sekunde. Časovnik pri tome ne prekida sa radom.

Ako je potrebno čitati samo jedan registar, ne postoji problem prenosa, tj. nije potrebno privremeno pamćenje.

Ukoliko je bit 7 u kontrolnom registru CRB na logičkoj jedinici, časovnik je spreman za programiranje alarma. Alarm registri se nalaze na istim adresama kao i registri časovnika ali se u njih podaci mogu samo upisati (kod registra časovnika se može i upisivati i čitati). Kada časovnik odbroji do programirane vrednosti alarma, generiše se zahtev za prekidom.

Registar za kontrolu prekida (ICR)

Postoji pet mogućih izvora koji mogu generisati zahtev za prekidom:

- završetak brojanja tajmera A
- završetak brojanja tajmera B
- alarm iz časovnika realnog vremena
- serijski registar (SDR) pun (ako radi kao ulaz) ili prazan (ako radi kao izlaz).
- logička nula na izvodu $\overline{\text{FLAG}}$

Pri čitanju, ICR sadrži podatke o zahtevima za prekidom dok se pri upisivanju ICR ponaša kao registar maske. U suštini, fizički gledano, to su dva registra na istoj adresi. Pristup jednom registru je moguć samo u trenucima kada je $R/\overline{W}=1$, a drugom, kada je $R/\overline{W}=0$.

pri čitanju: (podaci o zahtevima za prekidom)

lokalna adresa	ime								
SD	ICR	IR	0	0	FLG	SP	ALRM	TB	TA

pri upisivanju: (maska prekida)

lokalna adresa	ime								
SD	ICR	$S/\overline{C}$	×	×	FLG	SP	ALRM	TB	TA

Svaki izvor prekida može da postavi odgovarajući bit na logičku jedinicu kao svoj zahtev. Ukoliko je maskom dozvoljen prekid, postaviće se IR (opšti zahtev) na logičku jedinicu, a linija $\overline{\text{IRQ}}$ će se postaviti na nulu.

U sistemima sa više CIA, IR se ispituje za vreme prozivke posle nastalog prekida da bi se utvrdilo koji CIA je generisao prekid. Po čitanju podataka iz ICR, podaci se brišu, a $\overline{\text{IRQ}}$ se postavlja na jedinicu pa ove podatke treba negde privremeno smestiti u memoriji. Ovde treba uočiti da se podaci zahteva za prekidom uvek upisuju u ICR bez obzira na masku. Pri tome maska omogućuje samo pojedinim zahtevima da generišu opšti zahtev (IR i $\overline{\text{IRQ}}$).

Prema tome, posle sekvence prozivke utvrđuje se koji CIA je generisao prekid, smešta se sadržaj ICR registra privremeno u memoriju, a zatim sledi nova prozivka da bi se utvrdio konkretan izvor prekida unutar CIA. Pri tome se softverski dodeljuju prioriteti ukoliko je više prekida generisano istovremeno.

Kada se upisuje u ICR (registar maska), ako je bit $S/\overline{C}$ (set/$\overline{\text{clear}}$) na logičkoj nuli, svaki bit maske koji je poslat kao jedinica biće u stvari obrisan (upisana logička nula), dok poslata logička nula neće promeniti stanje odgovarajućeg bita (operacija $C = A\ \overline{B}$). Ukoliko je $S/\overline{C}$ na logičkoj jedinici, svaki bit maske poslat kao jedinica biće upisan kao jedinica (setovan), dok poslata logička nula neće promeniti stanje (operacija $C = A + B$).

Da bi bilo omogućeno generisanje opšteg zahteva za prekidom (IR), odgovarajući bit u maski mora da bude na logičkoj jedinici (setovan).

Primer: IRC

	0	X	X	1	0	1	1	0	A
write	0	X	X	0	0	1	0	0	B
rezultat	0	X	X	1	0	0	1	0	$C = A.\overline{B}$
write	1	X	X	0	1	0	1	0	D
rezultat	1	X	X	1	1	0	1	0	$E = C + D$

Funkcije pojedinih izvoda CIA 6526

GND, masa

PA0 do PA7, dvosmerne linije podataka za kapiju A (ulazi ili izlazi)

$\overline{\text{PC}}$, kontrolna linija, ulaz

Odnosi se samo na kapiju B. Ako je kapija B ulazna (DDRB = $00), logička nula na PC signalizira da su podaci spremni za učitavanje (engl. data ready). Ako je kapija B izlazna (DDRB = $FF), logička nula na PC signalizira da su podaci pročitani sa kapije B (engl. data accepted). Ako se ostvaruje šesnaestobitni prenos podataka (koristeći kapije A i B), uvek se prvo obraća kapiji A.

TOD, ulaz, izvod za sinhronizaciju časovnika realnog vremena (50 Hz)

Vcc, napajanje +5V

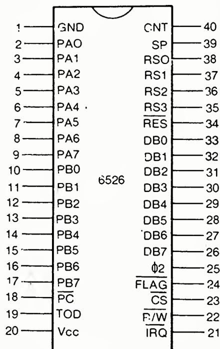
$\overline{\text{IRQ}}$, (engl. interupt request), izlaz

Ovaj izvod je normalno priključen na $\overline{\text{IRQ}}$ procesorsku liniju. On je sa otvorenim drejnom što omogućuje da se više čipova preko zajedničkog otpornika prema plus polu napajanja (engl. pull up) priključi na zajedničku liniju. Dolazi u stanje logičke nule kada se u CIA generiše opšti zahtev za prekidom.

$R/\overline{W}$, (engl. read/write), ulaz

Procesorska kontrolna linija za određivanje da li se vrši upisivanje ili čitanje.

Sl. 11. 31. Raspored izvoda na kolu 6526



$\overline{CS}$, ulaz

Logička nula omogućuje pristup pojedinim registrima u trenutku aktivnog takta Φ2 i odgovarajuće kombinacije adresa RS3 do RS0.

$\overline{FLAG}$, ulaz

Ovaj ulaz je osetljiv na negativnu ivicu, pri čemu generiše zahtev za prekidom postavljajući na logičku jedinicu bit FLG u IRC. Može se koristiti za primanje informacija sa izvoda $\overline{PC}$ nekog drugog CIA ili kao univerzalni ulaz za generisanje prekida. U kombinaciji sa $\overline{PC}$ formira par linija za kontrolu toka podataka (engl. handshake lines).

Φ2, ulaz, sistemski takt

DB0 do DB7, procesorska sistemska magistrala podataka

Dvosmerne linije (ulazi i izlazi).

$\overline{RES}$, ulaz

Logička nula na ovom ulazu resetuje sve unutrašnje regištre. Kapije se definišu kao ulazne, serijski izvod kao ulazni a u odgovarajuće registre se upisuju nule (PRA, PRB, SDR = $00). Kontrolni registri svih tajmera se postavljaju na nulu, a tajmer lečevi na jedinice. Svi ostali registri se postavljaju na vrednost nula.

RS0 do RS3, adresna magistrala, ulazi

Može da adresira 16 registara.

SP, dvosmerna linija serijskih podataka

U zavisnosti od toga da li je bit SPMODE 0 ili 1 određuje se da li se linija ponaša kao ulaz ili izlaz.

CNT, linija za sinhronizaciju serijskih podataka.

Dvosmerna linija (ulaz ili izlaz) u zavisnosti od toga da li je SP ulaz ili izlaz, kao i od vrednosti bita INMODE.

11.6.5. Tastatura

Tastatura je organizovana kao matrica 8 vrsta puta 8 kolona. Ona je povezana na CIA 1. Kapija A je programirana kao izlaz, a kapija B kao ulaz. Tajmer A i B su programirani kao monostabilna kola. Tajmer A je programiran da broji tačno 16421 ciklusa takta Φ2 posle čega generiše zahtev za prekidom preko linije $\overline{IRQ}$. To se događa svakog šezdesetog dela sekunde (16421/Φ2 = 1/60 sekunde).

Rutina za obradu prekida, pored ostalih poslova, zove potprogram SCNKEY ($EA87) čiji je zadatak čitanje tastature. Čitanje se obavlja ovako: CIA 1 postavlja izvod PA0 na logičku nulu, a zatim čita podatak preko kapije B. Ako je pritisnut neki taster u prvom redu, pojaviće se na odgovarajućoj liniji logička nula. U zavisnosti od toga na kom težinskom mestu

Sl. 11. 32. Veza tastature sa CIA 1

je ta nula, skače se indirektno na dekodersku tabelu i uzima kôd (redni broj) tastera koji je pritisnut. Ovaj broj se stavlja u memorijsku lokaciju SFDX ($CB). Posle ovoga CIA postavlja PA0 na logičku jedinicu, a PA1 na logičku nulu pa se ceo postupak ponavlja. Ukoliko je još neki taster pritisnut, obaviće se isti postupak i njegov kôd će biti smešten u SFDX. Tako ide sve do osmog reda s tim što će taster sa najvećim rednim brojem biti smešten kao poslednji u SFDX.

Potprogram se završava stavljanjem kôda u bafer tastature, povećanjem broja karaktera u baferu za 1, što se događa na lokaciji NDX ($C6) i smeštanjem kôda tastera na lokaciju LSTX ($C5). Na ovoj lokaciji je smešten kôd poslednjeg pritisnutog tastera.

Čitanje sa tastature se zbog veće pouzdanosti obavlja na sledeći način (engl. debouncing).

```
READ    LDA PRB1
        CMP PRB1
        BNE READ
```

Prvo se učita bajt iz CIA 1 kapije B pa se njegova vrednost zatim uporedi sa trenutnom vrednošću kapije B da bi se utvrdilo da nije u međuvremenu došlo do neke smetnje ili izmene. Tek kada se poređenjem ustanovi da nema greške, nastavlja se dalje.

Kodovi tastera (redni brojevi) dati su u sledećoj tabeli:

taster	kod	taster	kod	taster	kod	taster	kod
A	10	Q	62	7	24	RETURN	1
B	28	R	17	8	27	.	47
C	20	S	13	9	32	.	44
D	18	T	22	0	35	/	55
E	14	U	30	+	40	↓↑	7
F	21	V	31		43	⇆	2
G	26	W	9	£	48	F1	4
H	29	X	23	CLR HOME	57	F3	5
I	33	Y	25	INST/DEL	0	F5	6
J	34	Z	12	−	57	F7	3
K	37	1	56	@	46	SPACE	60
L	42	2	59		49	RUN/STOP	63
M	36	3	8	↑	54	nijedan	64
N	39	4	11	:	45	CTRL	58
O	38	5	16	;	50	⇐	61
P	41	6	19	=	53	SHIFT LEFT	15
						SHIFT RIGHT	52

Kodovi tastera u kombinaciji sa pritisnutim tasterima SHIFT, CTRL i C= kao i pomoću odgovarajućih tabela u memoriji daju odgovarajuće PET ASCII kodove koji se dalje standardno mogu koristiti.

Taster RESTORE ima posebnu ulogu. Pomoću njega, otpornika R35, R41 i kondenzatora C38 okida se tajmer U20 koji radi kao monostabil. Trajanje izlaznog impulsa je određeno sa R33 i C23 (oko 18 μs). Izlazni impulsi se invertuju u kolu U8, a zatim vode na $\overline{NMI}$ ulaz mikroprocesora. Na taj način pritiskom na taster RUN/STOP i RESTORE, može se prekinuti trenutno izvršavanje programa (koji ne mora da bude u bejziku) i vratiti se u editor. Pri ovome se ekran briše,a kursor se postavlja u gornji levi ugao ispod poruke READY

11.6.6 IEEE 488 standard

Magistrala tipa IEEE 488 je nastala iz asinhronog sistema za razmenu podataka koji je prvi put uveo Hewlett-Packard (HPIB). 1975 je ovaj standard formalizovan i od tada se široko primenjuje pri povezivanju računara sa raznim periferijskim uređajima.

Uopšteno gledano, uređaj koji se nalazi priključen na magistralu 488 može biti u jednom od tri režima rada:

- učitavanje podataka sa magistrale – prijemnik (engl. listener)
- slanje podataka preko magistrale – predajnik (engl. talker)
- kontrola drugih uređaja – kontroler (engl. controler)

Priključeni uređaji mogu biti u mogućnosti da obavljaju bilo koju kombinaciju ove tri funkcije, s tim što na magistrali u jednom trenutku može da bude aktivan samo jedan kontroler.

Svakom uređaju je dodeljena jedinstvena adresa. Kontroler adresira proizvoljan broj prijemnika iako se u jednom trenutku informacija prenosi samo između dva uređaja: jednog predajnika i jednog prijemnika. Računar obično obavlja sve tri funkcije: kontrolu, predaju i prijem. Disk jedinica, na primer, može biti predajnik i prijemnik, dok je štampač samo prijemnik.

Preko magistrale 488 prenos se obavlja paralelno. Standardizovani su TTL logički nivoi sa negativnom logikom (aktivno stanje je logička nula).

Postoje tri tipa linija:

1. Linije podataka. To su dvosmerne linije ($\overline{DIO}$ 1 – 8) koje prenose jedan bajt adrese, podatka ili komande.

2. Linije za kontrolu razmene (handshake lines):

a) $\overline{DAV}$ (engl. data available). Kada se ova linija postavi forsirano na logičku nulu od strane predajnika, informaciju na magistrali mogu da pročitaju svi prijemnici.

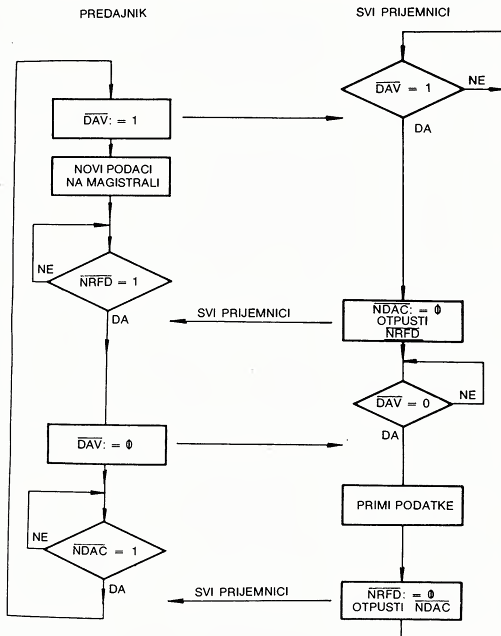
b) $\overline{NRFD}$ (engl. not ready for data). Svi prijemnici su povezani na ovu liniju direktno svojim izlazima koji su otvoreni kolektori i na taj način formiraju logičku ILI operaciju (engl. wired OR). Kada je prijemnik spreman za prijem podataka, on svoj izlaz otpušta u stanje logičke jedinice. Otpustiti liniju znači da se izlaz stavlja u stanje logičke jedinice čime se stvara uslov da, ako i ostali uređaji to učine, odgovarajuća linija bude na logičkoj jedinici. Prema tome, linija $\overline{NRFD}$ će biti na logičkoj jedinici kada svi prijemnici budu spremni za prijem podataka.

c) $\overline{NDAC}$ (engl. not data accepted). Svi prijemnici su povezani preko svojih izlaza sa otvorenim kolektorom na ovu liniju. Kada odgovarajući prijemnik primi podatke, on svoj izlaz otpušta u stanje logičke jedinice. Kada poslednji prijemnik primi podatak, $\overline{NDAC}$ se postavlja na logičku jedinicu što predajniku daje signal da može da ukloni podatak sa magistrale.

3. Linije za upravljanje.

a) $\overline{ATN}$ (engl. attention). Kontroler postavlja ovu liniju u stanje logičke nule stavljajući na taj način do znanja da je informacija koja sledi adresa ili komanda.

b) $\overline{IFC}$ (engl. interface clear). Kontroler može postaviti ovu liniju u stanje logičke nule bilo kada, ali to obično čini po uključenju uređaja. Posle ovoga svi uređaji se dovode u neutralno stanje u odnosu na magistralu u roku od 100 μs, pri čemu ne znači da se sami uređaji automatski reinicijalizuju (resetuju), mada može i to da se dogodi. Inicijalizacija se obično izvodi slanjem posebnog komandnog koda.



Sl. 11. 33. Blok dijagram protokola IEEE 488

c) $\overline{EOI}$ (engl. end or identify). Ovu liniju postavlja predajnik na logičku nulu označavajući poslednji bajt podatka koji šalje. $\overline{EOI}$ liniju može koristiti i kontroler u kombinaciji sa linijom $\overline{ATN}$. U tom slučaju, kada su i $\overline{ATN}$ i $\overline{EOI}$ na logičkoj nuli, obavlja se takozvana paralelna prozivka u kojoj mogu da učestvuju maksimalno 8 uređaja. Svaki od njih u tom trenutku na jednu od 8 linija podataka postavlja informaciju o svom statusu. Ovaj odziv mora uslediti u roku od 200 μs.
Drugi slučaj je identifikacija uređaja koji je postavio $\overline{SRQ}$.

Za vreme serijske prozivke uređaj postavlja na linije „DATA" svoju adresu.

d) $\overline{SRQ}$ (engl. service request). Bilo koji uređaj može tražiti od kontrolera da bude uslužen postavljajući ovu liniju u stanje logičke nule. Kontroler ovo tretira kao zahtev za prekidom (interrupt request), pri čemu određuje koji je uređaj poslao zahtev i to na osnovu jednog od sledećih podataka:
– unapred određenim rasporedom opsluživanja uređaja
– paralelnom prozivkom
– serijskom prozivkom

Serijska prozivka se odvija kao niz komandi i zahteva za statusom od jednog uređaja do drugog. Ona je sporija od paralelne prozivke, ali omogućuje ispitivanje kompletnog stanja uređaja.

e) $\overline{REN}$ (engl. remote enable). Ovu liniju koristi kontroler da bi omogućio upotrebu kontrola na prednjoj ploči nekog uređaja, obično nekog mernog instrumenta i sl. Kontrola se postavlja daljinski. Kada se ova linija postavi na logičku jedinicu, svi uređaji moraju u roku od 100 μs da pređu u stanje lokalne kontrole.

Komandne reči

Komandne reči (bajtove) šalje kontroler preko linija podataka $\overline{DIO1}$ –7 za vreme dok drži liniju $\overline{ATN}$ na logičkoj nuli. Linija $\overline{DIO8}$ se prema standardu pri tome ne koristi. Ona može biti upotrebljena zavisno od sistema ili korisnika (npr. za proveru parnosti ili slično).

1. x 0 1 A4 A3 A2 A1 A0 gde je A4A3A2A1A0≠11111 – listen. Svi uređaji prihvataju ovu komandu, ali samo onaj sa poslatom adresom postaje prijemnik.

2. x 0 1 1 1 1 1 1 – unlisten. Svi uređaji koji su bili prijemnici prestaju to da budu.

3. x 1 0 A4 A3 A2 A1 A0 gde je A4A3A2A1A0≠11111 – talk. Uređaj sa odgovarajućom adresom postaje predajnik. Prethodni predajnik prestaje to da bude. Na taj način se izbegava istovremeno prisustvo više predajnika.

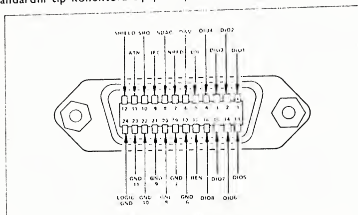
4. x 1 0 1 1 1 1 1 – untalk. Uređaj koji je bio predajnik prestaje to da bude. Uređaj ili instrumenti priključeni na 488 magistralu obično imaju mikro prekidače pomoću kojih se postavlja njihova adresa.

5. Univerzalne komande. Imaju dejstva na sve uređaje. Na primer x 0 0 1 0 1 0 0 – clear; resetuje sve uređaje u njihovo početno stanje.

6. Adresirane komande. Imaju dejstva samo na trenutne prijemnike. Na primer x 0 0 0 1 0 0 0 – group execute; startuje istovremene preprogramirane operacije svih prijemnika.

Raspored priključaka

IEEE 488 propisuje standardni tip konektora čiji je raspored izvoda sledeći:

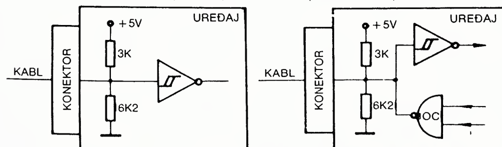


Sl. 11. 34. Izgled IEEE 488 priključka

pin	linija	pin	linija
1	$\overline{\text{DIO1}}$	13	$\overline{\text{DIO5}}$
2	$\overline{\text{DIO2}}$	14	$\overline{\text{DIO6}}$
3	$\overline{\text{DIO3}}$	15	$\overline{\text{DIO7}}$
4	$\overline{\text{DIO4}}$	16	$\overline{\text{DIO8}}$
5	$\overline{\text{EOI}}$	17	$\overline{\text{REN}}$
6	$\overline{\text{DAV}}$	18	DAV GND
7	$\overline{\text{NRFD}}$	19	NRFD GND
8	$\overline{\text{NDAC}}$	20	NDAC GND
9	$\overline{\text{IFC}}$	21	IFC GND
10	$\overline{\text{SRQ}}$	22	SRQ GND
11	$\overline{\text{ATN}}$	23	ATN GND
12	oklop (masa)	24	SIGNAL GND

Struje i završni otpori 488 magistrale

Ulazne impedanse uređaja su propisane standardom i iznose 3 Koma, obezbeđujući uniformnu impedansu na liniji, kao i povećanu imunost na smetnje. Izlazni sklop za pobuđivanje (drajver) mora da izdrži struju od 48 mA. Maksimalan broj uređaja treba ograničiti na 14, a dužinu kablova na 2m po uređaju s tim da ukupna dužina ne pređe 20m.



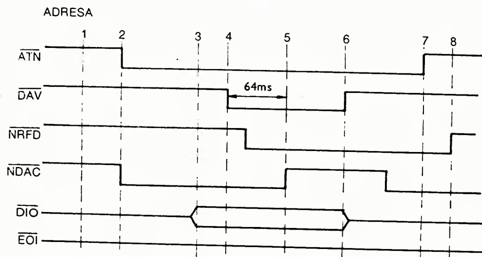
Sl. 11. 35. Standardne otporne mreže za: a) unidirekcione linije b) bidirekcione linije

Vremenski dijagrami

I Adresiranje

1. Sve linije su na nivou logičke jedinice osim linija podataka koje su u stanju visoke impedanse.

2. Kontroler postavlja liniju ATN na logičku nulu stavljajući do znanja svim uređajima da sledi komanda. Tada svi uređaji postavljaju svoj NDAC izlaz na logičku nulu.



Sl. 11. 36. Vremenski dijagrami pri adresiranju kod IEEE 488 veze

3. Kontroler na linije podataka postavlja adresu i komandu.

4. Kontroler postavlja DAV liniju na logičku nulu označavajući da su podaci na linijama podataka valjani.

5. Adresirani uređaj mora u roku od 64ms da prihvati komandu i da to označi otpuštanjem linije NDAC na logičku jedinicu. Neposredno pre toga on postavlja NRFD na nulu, označavajući da je trenutno zauzet (nespreman za nove podatke). Ukoliko sve ovo ne uradi, dolazi do greške DEVICE NOT PRESENT.

6. Po prijemu signala NDAC=1, kontroler postavlja DAV na logičku jedinicu označavajući da podaci na magistrali nisu valjani.

7. Kontroler postavlja ATN=1 završavajući komandnu poruku.

8. Svi uređaji otpuštaju liniju NRFD, označavajući spremnost za prijem novih podataka.

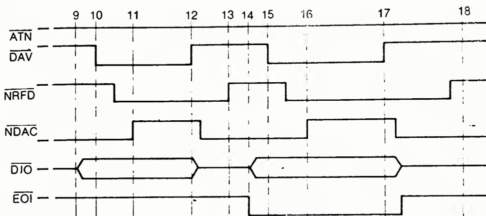
II Razmena podataka

Pošto je kontroler odredio uloge pojedinih uređaja, može doći do razmene podataka između predajnika i prijemnika. Najčešće sam kontroler (računar) ima jednu od ove dve funkcije.

9. Stanje po završetku prethodne operacije (bila ona adresiranje ili razmena podataka). Predajnik stavlja podatak na magistralu.

10. Predajnik, postavljajući liniju DAV na nivo logičke nule, označava da su podaci valjani.

11. Prijemnik prvo postavlja NRFD=0. Na taj način javlja predajniku da čeka sa slanjem novih podataka. Zatim, otpuštanjem linije NDAC, javlja da je podatak primio.



Sl. 11. 37. Vremenski dijagrami pri razmeni podataka kod IEEE 488 veze

12. Predajnik postavlja $\overline{\text{DAV}}=1$ označavajući da podaci više nisu valjani.

13. Svi prijemnici su primili podatke i otpustili su liniju $\overline{\text{NRFD}}$ pa ona prelazi u stanje logičke jedinice.

14. Stanje je isto kao i u trenutku 9. Ukoliko predajnik namerava da pošalje poslednji podatak, on postavlja $\overline{\text{EOI}}=0$. Posle ovoga sledi normalan prenos poslednjeg podatka pri čemu stanje 15 odgovara onom pod 10. Na isti način se uspostavlja korespondencija između stanja 16 i 11, 17 i 12, 18 i 13 pri čemu se završava razmena podataka.

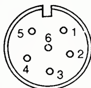
10.6.7 Komodorova serijska veza (IEC)

Kod Komodora su implementirane sve funkcije po standardu IEEE 488, ali se one obavljaju na drugačiji način.

Podaci se prenose serijski, bit po bit, dok se kontrolne operacije obavljaju u vremenskom multipleksu.

Ovom serijskom vezom se povezuju disk jedinice, štampači, ploteri. Istovremeno može biti prisutno 5 uređaja.

1 – $\overline{\text{SRQ}}$
2 – $\overline{\text{GND}}$
3 – $\overline{\text{ATN}}$
4 – $\overline{\text{CLK}}$
5 – $\overline{\text{DATA}}$
6 – $\overline{\text{RESET}}$



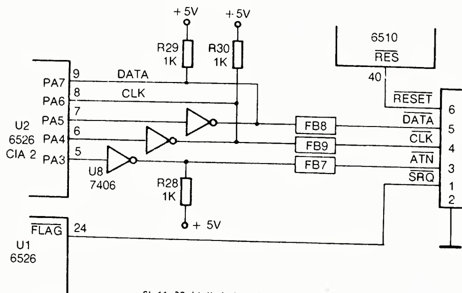
Sl. 11. 38. Izgled konektora za serijsku vezu

Kontroler je isključivo sam Komodor 64.

Linije $\overline{\text{SRQ}}$ i $\overline{\text{ATN}}$ imaju istu funkciju kao i kod IEEE 488 interfejsa. Preko nilija $\overline{\text{CLK}}$ i $\overline{\text{DATA}}$ se obavlja kompletna komunikacija. Linija $\overline{\text{RESET}}$ se direktno vodi na $\overline{\text{RES}}$ izvod mikroprocesora 6510.

Kompletan protokol je izveden softverski i deo je operativnog sistema. Hardver čin CIA 1 i 2 kao i odgovarajući invertori/drajveri.

Vidi se da je $\overline{SRQ}$ linija vezana na $\overline{FLAG}$ ulaz kola U1 (CIA 1), tj. direktno za istu tačk na koju je vezana i linija CASSRD.

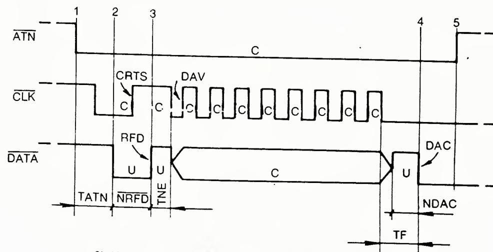


Sl. 11. 39. Linije koje se koriste za serijsku vezu

Vremenski dijagrami

Funkcionisanje Komodorove IEC veze najbolje se može opisati vremenskim dijagra mima.

I Adresiranje



Sl. 11. 40. Vremenski dijagrami pri adresiranju kod serijske veze

Na dijagramu postoje sledeće oznake:
RFD – ready for data
$\overline{\text{NRFD}}$ – not ready for data
DAC – data accepted
NDAC – not data accepted
C – Komodorov signal
U – signal sa uređaja
T – signal sa predajnika
L – signal prijemnika

1. Komodor postavlja liniju $\overline{\text{ATN}}$ u stanje logičke nule. U intervalu TATN od maksimalno 1ms mora doći do odziva uređaja u vidu obaranja linije $\overline{\text{DATA}}$ na logičku nulu. U ovom intervalu Komodor postavlja i $\overline{\text{CLK}}$ na nulu.

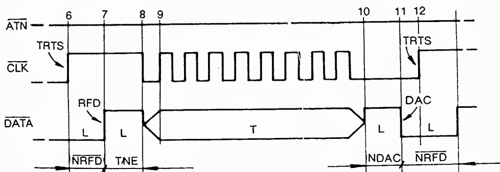
2. Linija $\overline{\text{DATA}}$ preuzima funkciju $\overline{\text{NRFD}}$. U međuvremenu Komodor postavlja $\overline{\text{CLK}}$ na logičku jedinicu označavajući da je spreman za slanje podataka – CRTS (engl. controler ready to send).

3. Svi uređaji otpuštaju liniju $\overline{\text{DATA}}$ i čekaju podatke. Podaci se šalju serijski, s tim što prvo ide MSB, a na kraju LSB. Podaci su valjani u trenucima prelaska $\overline{\text{CLK}}$ na logičku jedinicu, tj. na prednjoj ivici (funkcija DAV).
Posle 8 prenetih bita, u roku od TF=1ms (maksimalno), adresirani uređaj mora spustiti liniju $\overline{\text{DATA}}$ označavajući da je podatak primljen (funkcija NDAC).

4. Kada dobije potvrdu da je podatak primljen, Komodor postavlja $\overline{\text{ATN}}$ na logičku jedinicu.

II Razmena podataka

6. Predajnik postavlja liniju $\overline{\text{CLK}}$ na logičku jedinicu, što znači da je spreman za slanje podataka – TRTS (engl. talker ready to send). Za to vreme prijemnici drže liniju $\overline{\text{DATA}}$ na logičkoj nuli (funkcija NRFD).



Sl. 11. 41. Vremenski dijagrami pri razmeni podataka kod serijske veze

7. Kada su svi prijemnici spremni za prijem podatka, oni otpuštaju liniju $\overline{\text{DATA}}$, koja na kraju prelazi u stanje logičke jedinice.

8. U intervalu TNE od maksimum 200 μs, predajnik mora da postavi $\overline{\text{CLK}}$ na nulu da bi označio početak slanja poruke. Ukoliko je ovo vreme veće, radi se o slanju poslednjeg bajta (funkcija EOI).

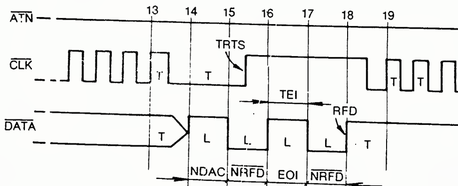
9. Na pozitivnoj ivici signala na liniji $\overline{\text{CLK}}$ podaci su valjani (DAV).

10. Posle 8 poslatih bita, prijemnik drži $\overline{DATA}$ liniju na logičkoj jedinici (funkcija NDAC) maksimalno 1ms.

11. Negativnom ivicom na liniji $\overline{DATA}$ prijemnik ukazuje predajniku da je podatak prihvaćen (DAC) ulazeći u stanje $\overline{NRFD}$.

12. Predajnik ponovo postavlja $\overline{CLK}$ na jedinicu i tako se ceo ciklus ponavlja sve do poslednjeg podatka.

III Razmena poslednjeg podatka



Sl. 11. 42. Vremenski dijagrami pri razmeni poslednjeg podatka kod serijske veze

13. Prenos poslednjeg bita poslednjeg podatka.

14. Prijemnik funkcija NDAC, maksimalno 1ms.

15. Prijemnik funkcija $\overline{NRFD}$.

16. Predajnik postavlja $\overline{CLK}$ na jedinicu, a prijemnik označava da je spreman za prijem podataka otpuštajući liniju $\overline{DATA}$, posle čega pošto u roku od TEI=200 µs predajnik nije postavio $\overline{CLK}$ na nulu, prijemnik zna da je prethodni podatak bio poslednji.

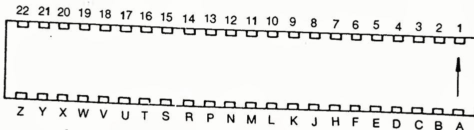
17. Prijemnik označava da je prihvatio poruku o poslednjem podatku, postavljajući $\overline{DATA}$ liniju na logičku nulu tj. vraćajući se u stanje $\overline{NRFD}$.

18. Prijemnik postavlja $\overline{DATA}$ na jedinicu ukazujući predajniku da je spreman za prijem osam nula ($00).

19. Predajnik šalje prvu od osam uzastopnih nula.

11.6.8. Priključak za proširenja

Na priključak za proširenja su izvedene adresne linije, linije podataka i linije potrebne za rad dodatog uređaja.



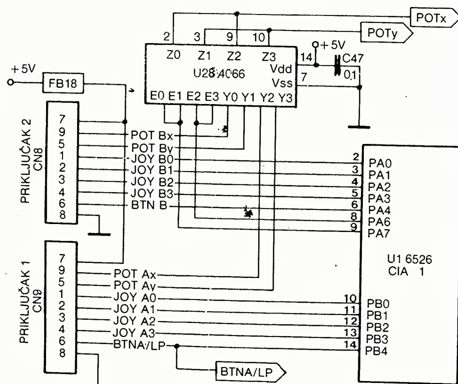
Sl. 11. 43. Izgled priključka za proširenja. Pogled sa zadnje strane računara

ime	izvod	opis
GND	1	masa
+5V	2	napajane, priključeni uređaj
+5V	3	maksimalna potrošnja 450mA
$\overline{IRQ}$	4	IRQ linija mikroprocesora
R/$\overline{W}$	5	R/$\overline{W}$ linija mikroprocesora
DOT CLOCK	6	7.88198MHz video takt
$\overline{I/O1}$	7	ulazno izlazni blok 1 ($DE00 – $DEFF) nebaferisan
$\overline{GAME}$	8	ulazi za signaliziranje Kernalu da postoji
$\overline{EXROM}$	9	spoljašnji ROM
I/O2	10	ulazno izlazni blok 2 ($DF00 – $DFFF) baferisan
$\overline{ROML}$	11	dekodovani RAM/ROM blok od 8K na $8000, baferisan
BA	12	BA signal iz video kontrolera
$\overline{DMA}$	13	zahtev za direktnim pristupom memoriji
D7	14	
D6	15	
D5	16	
D4	17	magistrala podataka
D3	18	
D2	19	
D1	20	
D0	21	
GND	22	masa
GND	A	masa
$\overline{ROMH}$	B	dekodovani RAM/ROM blok od 8K na $E000
$\overline{RESET}$	C	RES linija mikroprocesora
$\overline{NMI}$	D	NMI linija mikroprocesora
Φ2	E	sistemski takt
A15	F	
A14	H	
A13	J	
A12	K	
A11	L	
A10	M	
A9	N	
A8	P	adresna magistrala
A7	R	
A6	S	
A5	T	
A4	U	
A3	V	
A2	W	
A1	X	
A0	Y	
GND	Z	masa

11.6.9. **Upravljački ulazi**

Komodor poseduje dva upravljačka priključka (engl. game port) na koje se mogu priključiti analogna ili digitalna palica kao i svetlosna olovka (engl. joystick, paddle, light pen).

Svetlosna olovka se može priključiti samo na priključak br. 1. Analogne palice mogu imati po dva potenciometra, a na elektronskom preklopniku U28 tipa 4066 bira se koji par će biti prosleđen na A/D konvertor u SID integrisanom kolu. Očitavanje vrednosti iz A/D konvertora je neprecizno ako se obavlja iz bejzika pa se za ovo preporučuje upotreba mašinskog potprograma.

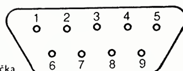


Sl. 11. 44. Povezivanje upravljačkih ulaza

Linije sa digitalnih palica vode se direktno na izvode periferijskog integrisanog kola CIA 1. U programu koji koristi palice, ovi izvodi mora da budu programirani kao ulazni. PA6 i PA7 se programiraju kao izlazni i koriste se za izbor jedne od analognih palica (pot. Ax, Ay ili pot. Bx, By).

Vidi se da su linije priključka 1 vezane na izvode CIA 1 na koje su takođe vezane linije iz tastature. Zbog toga se neke funkcije tastature mogu imitirati palicom kao i obratno. Ovo ne važi za priključak 2 jer su izvodi CIA 1, koji su paralelno spojeni sa linijama tastature i linijama palice, programirani kao izlazni u slučaju tastature, a kao ulazni u slučaju palice.

BTNA i BTNB su linije prekidača za paljbu, s tim što se BTNA vodi na izvod LP video kontrolera u slučaju priključenja svetlosne olovke.



Sl. 11. 45. Izgled upravljačkog priključka

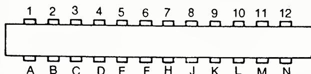
priključak izvod	1 tip	komentar
1	JOYA0	gore
2	JOYA1	dole
3	JOYA2	desno
4	JOYA3	levo
5	POT AY	
6	BTNA/LP	paljba/LP
7	+5V	max. 50mA
8	POT AX	

izvod	tip	komentar
1	JOYB0	gore
2	JOYB1	dole
3	JOYB2	desno
4	JOYB3	levo
5	POT BY	
6	BTNB	paljba
7	+5V	max. 50mA
8	POT BX	

11.6.10 **Korisnički priključak**

Korisnički priključak (engl. user port) služi za povezivanje Komodora sa spoljnim svetom. Priključak je povezan sa izvodima integrisanog kola CIA 2 koji mogu biti ulazni i izlazni. Na taj način se Komodor može povezati sa printerom, modemom ili nekim specijalnim uređajem. Normalno, za svaku od ovih funkcija CIA 2 se mora posebno programirati.

Sl. 11. 46. Povezivanje CIA 2 na korisnički priključak



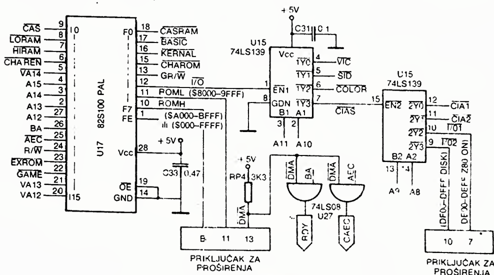
Sl. 11. 47. Izgled korisničkog priključka

gornja strana	
izvod	tip
1	GND
2	+5V (100mA max)
3	RESET
4	CNT1
5	SP1
6	CNT2
7	SP2
8	PC2
9	ATN
10	9VAC (50mA max)
11	9VAC (50mA max)
12	GND

donja strana	
izvod	tip
A	GND
B	FLAG
C	PB0
D	PB1
E	PB2
F	PB3
H	PB4
J	PB5
K	PB6
L	PB7
M	PA2
N	GND

11.7 LOGIKA ZA UPRAVLJANJE MEMORIJOM

Kao što se može videti iz organizacije memorije, Komodor poseduje 64 KB RAM-a i 20 KB ROM-a što je više od adresnog prostora mikroprocesora 6510. Tu su, zatim, dodatni spoljašnji ROM moduli, Z80 (CP/M) moduli, i razni ulazno-izlazni dodaci koji zauzimaju lokacije u memorijskom prostoru. Osim mikroprocesora 6510, i video kontroler ima zahteve za pristupom memoriji.



Sl. 11. 48. Logika za upravljanje memorijom

Da bi se pojedini delovi memorije u pravom trenutku dodelili mikroprocesoru ili video kontroleru, koristi se logika za upravljanje memorijom MMU (engl. Memory management unit). Ona obavlja dekodovanje adresa i multipleksiranje sa kontrolnim signalima.

Logički sklop se sastoji od dva integrisana kola:
1. U17 tipa 82S100. To je PAL (engl. programmable array logic) tj. programabilna logička mreža. Ona u sebi sadrži veći broj I, ILI i NE kola koja su povezana na određeni način i čine kombinacionu mrežu. Na taj način je izbegnuto korišćenje desetak standardnih integrisanih kola.
2. U15 tipa 74LS139. To je dvostruki dekoder 1 od 4 i koristi se za adresiranje ulazno—izlaznih integrisanih kola za vreme dok je aktivna linija I/O. Za adresiranje se koriste linije A11 do A8.

Adresni prostori koji zauzimaju pojedine jedinice su

		A11	A10
$\overline{\text{VIC}}$	SD000 — SD3FF	0	0
$\overline{\text{SID}}$	SD400 — SD7FF	0	1
$\overline{\text{COLOR}}$	SD800 — SDBFF	1	0
$\overline{\text{CIAS}}$	SDC00 — SDFFF	1	1

Signal $\overline{\text{CIAS}}$ se vodi na $\overline{\text{EN2}}$ (engl. enable), ulaz drugog dekodera integrisanog kola U15, pa se u adresnom prostoru SDC00 — SDFFF dobijaju sledeći signali:

		A11	A10	A9	A8
CIA1	SDC00 — SDCFF	1	1	0	0
CIA2	SDD00 — SDDFF	1	1	0	1
I/01	SDE00 — SDEFF	1	1	1	0
I/02	SDF00 — SDFFF	1	1	1	1

Očigledno je da ulazno/izlazne jedinice (tj. njihovi registri) ne zauzimaju ceo dodeljeni adresni prostor. Adrese koje nisu zauzete, a pripadaju dodeljenom adresnom prostoru, nazivaju se slike (engl. images) odgovarajućih adresa. Tako, na primer, adrese SD500 — D5FF, SD600 — SD6FF i SD700 — SD7FF predstavljaju slike adresa SD400 — SD4FF na kojima se realno nalaze registri SID integrisanog kola.

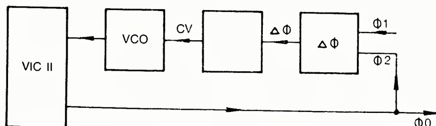
Upravljanje mikroprocesorom se obavlja pomoću I kola U27 tipa 74LS08. Signali AEC i BA, koje daje video kontroler, normalno se vode na mikroprocesor ukoliko je linija $\overline{\text{DMA}}$ na logičkoj jedinici. Ukoliko neka spoljašnja jedinica uputi zahtev za direktnim pristupom memoriji ($\overline{\text{DMA}}=0$), linije AEC i RDY trenutno prelaze u stanje logičke nule što dovodi do odvajanja mikroprocesora od adresne magistrale i magistrale podataka.

11.8 LOGIKA ZA GENERISANJE TAKTOVA

Svi takt-signali u celom sistemu se dobijaju obradom signala iz jednog jedinog oscilatora. Taj oscilator čine integrisano kolo U1 tipa 74LS629, kvarc Y1, kondenzator C70 i otpornici R26 i R27. Ovo je naponsko kontrolisani oscilator (engl. VCO — voltage controlled oscillator) čija je osnovna frekvencija oscilovanja 17.734472MHz. Usled tolerancije kvarca, ova frekvencija može malo da odstupi od tačne vrednosti, što se ispravlja kontrolnim naponom koji se dobija sa trimer potenciometra R27. Sa izlaza VCO, impulsi se vode direktno na Φ COLOR ulaz video kontrolera. Iz ovoga se vidi zašto frekvencija mora da bude vrlo tačna. Naime Φ COLOR signal se u video kontroleru deli sa 4 i na taj način se dobija noseći signal boje.

Sl. 11. 49. Logika za generisanje taktova

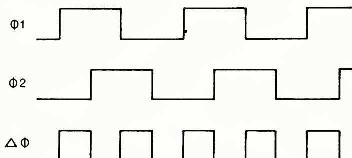
Osim ovoga, signal sa VCO izlaza se vodi na programabilni brojač U30 tipa 74LS193. On je programiran da broji na gore (engl. up counter) i da radi kao delitelj sa 9 u slučaju PAL TV sistema (kratkospajač J je u položaju E1 – E2). Tako se na njegovom izlazu dobija signal frekvencije 1.9704MHz, koji se odmah zatim deli sa 2 pomoću kola U29 (74LS74) koje radi kao T flip–flop.



Sl. 11.50. Princip rada PLL

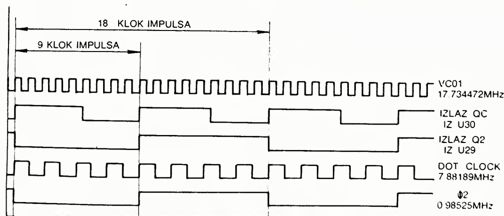
Integrisana kola U31, U32, VIC II i tranzistor Q7 čine takozvanu fazno sinhronizovanu petlju (engl. phase locked loop — PLL).

Dva signala bliskih frekvencija dovode se na fazni komparator U32. Na njegovom izlazu se dobija širinsko modulisan signal čija je širina proporcionalna faznoj razlici Φ1 — Φ2 što je akvivalentno operaciji ekskluzivno ILI.



Sl. 11. 51. Fazni odnos signala u pojedinim tačkama PLL

Dobijeni signal se filtrira niskopropusnim filtrom koga čine tranzistor Q7, otpornici R17, R19 i kondenzator C85. Na taj način se dobija jednosmerni napon proporcionalan faznoj razlici Φ1 — Φ2. Ovaj napon se koristi za kontrolu frekvencije VCO (U31) u nekoj oblasti Δf oko njegove sopstvene frekvencije f_0 (koja je fiksno podešena pomoću C86, R22 i R23). U ovom slučaju f_0 je podešena tačno na 7.88198MHz što je ujedno i sinhro signal za VIC II (DOT CLOCK). Video kontroler ovaj signal deli sa 8 tako da se na njegovom izlazu Φout dobija 0.98525MHz tj. sistemski takt. On se ponovo vodi na ulaz faznog komparatora i time se zatvara petlja.



Sl. 11. 52. Vremenski dijagrami u pojedinim tačkama

Ovako primenjen PLL omogućuje odličnu sinhronizaciju svih signala i vrlo visoku tačnost i stabilnost njihovih frekvencija.

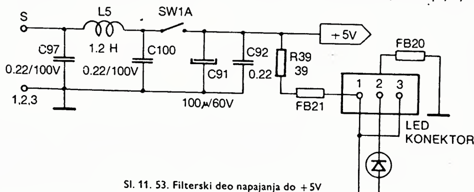
11.9 NAPAJANJE

Kolo za napajanje je podeljeno u dva dela. Jedan deo se nalazi u zasebnoj kutiji, ispravljaču, a drugi deo je u samom računaru.

Iz ispravljača se u računar, preko DIN priključka na njegovoj bočnoj strani, dovode dva napona:

stabilisani jednosmerni napon +5V
naizmenični napon 9Veff

Stabilisani napon od +5V koristi se za napajanje većine digitalnih integrisanih kola unutar računara. On se dovodi sa izvoda broj 5 (vrući kraj) i izvoda 1,2 i 3 (masa) DIN priključka. Napon se vodi na PI filter koji čine L5, C97 i C100. On filtrira impulsne smetnje koje se mogu javiti u vodovima za napajanje. Preko prekidača SW1A i niskopropusnog filtra C91, C92 (koji sprečava oscilovanje) napon se vodi na napajanje pojedinih integrisanih kola. U blizini izvoda za napajanje nekih integrisanih kola u računaru vrši se rasprezanje ovog napona keramičkim kondenzatorom od 0.1 do 0.47 uF. To se primenjuje zbog sprečavanja oscilovanja kao i zbog smanjenja uticaja jednog kola na drugo preko napona napajanja.



Sl. 11. 53. Filterski deo napajanja do +5V

Naponom +5V preko otpornika R39 napaja se i LED indikator napajanja. FB20 i FB21 su feritne perlice koje sprečavaju pojavu smetnji na radio frekvencijama.

Naizmenični napon od 9V se preko osigurača F1 i prekidača SW1B dovodi na diodni most u Grecovom spoju. Naravno, napon se prethodno filtrira od impulsnih smetnji filterom koji čine kondenzatori C20, C21, C98, C99 i spregnuti kalemovi L4.

Dobijanje ostalih potrebnih napona:

+5Vcan Dobija se filtriranjem ispravljenog (pomoću CR4) naizmeničnog napona od 9V. Filterski kondenzator je C19 dok C95 služi za njegovo premošćenje na višim frekvencijama jer ima manju unutrašnju impedansu. Napon se zatim stabilizuje na +5V regulatorom VR2. Kondenzatori C102 i C103 sprečavaju oscilovanje regulatora i poboljšavaju njegove tranzijentne osobine. Naponom +5Vcan napaja se video kontroler.

+Vvid Ovaj napon se dobija filtriranjem napona +5Vcan pomoću PI filtera koji čine L1, C61, C63, C64. Njime se napajaju video izlazni stepeni tj. hrominentni i luminentni pojačavač. Pošto ova kola rade na visokim frekvencijama, potrebno je ovo odvajanje pomoću PI filtera da bi se smanjio međusobni uticaj.

+Vc dobija se filtriranjem napona +5Vcan pomoću PI filtra L2, FB19, C65, C66, C67. Filtriranje se obavlja iz istih razloga kao i kod +Vvid. Ovim naponom se napaja logika za generisanje taktova.

Sl. 11. 54. Napajanje

+12V Pomoću udvajača napona koji čine CR5, CR6, C90 i C88 dobija se od 9V naizmeničnog napona jednosmeran napon od oko 16V. Ovaj napon se stabilizuje regulatorom VR1 na +12V. Kondenzatori C89, C57 i C59 imaju istu funkciju kao i kod VR2. Ovim naponom se napajaju video i audio kontroleri kao i tranzistor Q8 (audio izlazni stepen).

Vcas Ovaj napon se direktno vodi na elektroniku za upravljanje motorom kasetofona.

Vtod Ovo nije napon za napajanje nekog elektronskog sklopa, već je to signal sinhro impulsa koji se dovodi na TOD izvode kola CIA 1 i CIA 2. On je frekvencije 50Hz i dobija se iz mrežnog napona na sledeći način:

Preko otpornika R5 naizmenični napon 9V se dovodi na jedan ulaz I kola U27. Drugi ulaz je vezan na +5V. U toku pozitivne poluperiode, zener dioda CR1 je inverzno polarisana pa drži napon na ulazu na 2.7V. Ovo je dovoljno da I kolo na izlazu dâ logičku jedinicu, tj. +5V. U toku negativne poluperiode CR1 je direktno polarisana pa na ulazu drži napon od −0.7V što na izlazu daje logičku nulu tj. OV. Na ovaj način se vrši i zaštita kola U27 od negativnog i velikog pozitivnog napona. Otpornik R37 čini pozitivnu reakciju koja ubrzava prelazni režim, ali unosi i histerezis. Izlazni impulsi su, prema tome, četvrtke amplitude 5V, ali nejednakog odnosa impuls — pauza (što ovde nije od nekog značaja).

Na kraju treba napomenuti da su naponi +5V dostupni i preko priključka za proširenje (pin br. 2 i 3) i preko korisničkog priključka (pin br. 2).

Naizmenični napon od 9V je izveden samo na korisnički priključak (pin br. **10** i **11**).

12
Konstrukcije

U ovom poglavlju će biti prikazano nekoliko konstrukcija koje će upotpuniti mogućnosti primene računara Komodor 64.

Na osnovu datih objašnjenja i prikazanih šema moguće je realizovati dva standardna interfejsa za povezivanje računara sa štampačem, drugim računarima, modemom i drugim elektronskim uređajima.

Takođe je prikazan modem – sve aktuelniji uređaj za povezivanje dva računara preko telefonskih linija.

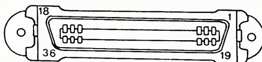
Potreba za jednostavnim i brzim učitavanjem programa koji se često koriste je dovela do realizacije ROM modula i EPROM programatora.

Za realizaciju navedenih uređaja je potrebno dobro poznavanje hardvera i softvera, kao i praktično iskustvo u gradnji elektronskih uređaja. U protivnom je moguće da uređaji ne funkcionišu ili čak da dođe do ozbiljnih oštećenja uređaja ili računara.

12.1 CENTRONIKS INTERFEJS

Centroniks je paralelni osmobitni interfejs koji služi za povezivanje perifernih uređaja sa računarom. Najčešće se koristi za slanje podataka na štampač. On ima osam linija podataka (DO – D7), STROBE liniju preko koje se šalje impuls kojim se podaci upisuju u periferni uređaj, BUSY liniju kojim periferni uređaj logičkom jedinicom obaveštava da je zauzet i nije spreman za prijem novih podataka i GND – uzemljenje. Centroniks standard propisuje i liniju ACK (engl. acknowledge – potvrda) kojom periferni uređaj potvrđuje prijem podataka ali se retko upotrebljava.

Kod Komodora je Centroniks interfejs lako implementirati jer postoji kompletna hardverska podrška. Potrebno je samo napraviti program i odgovarajući kabl. Za komunikaciju se koristi CIA 2 do koje se dolazi preko korisničkog priključka (engl. user port). Podaci se prenose preko kapije B dok se PA2 iz kapije A koristi za slanje STROBE impulsa. BUSY signal se dovodi na FLAG izvod.



Sl. 12. 1. Izgled Centonik s priključka

Centroniks konektor		korisnički priključak	
broj izvoda	naziv	naziv	broj izvoda
1	STROBE	PA2	M
2	D0	PB0	C
3	D1	PB1	D
4	D2	PB2	E
5	D3	PB3	F
6	D4	PB4	H
7	D5	PB5	J
8	D6	PB6	K
9	D7	PB7	L
10	ACK		
11	BUSY	FLAG 2	B
16	GND	GND	A

U inicijalizacionom deli program CIA 2 se programira da radi po Centroniks protokolu. Zatim sledi izmena vektora Kernal rutine za ispisivanje na izlaznu jedinicu. Nova rutina za ispisivanje je NEWCHR. U njoj se ispituje da li je izlazna jedinica broj 4 (štampač). Ukoliko nije, vraća se u staru CHROUT rutinu. U suprotnom se podatak šalje na izlaz uz generisanje STROBE impulsa. Ukoliko je BUSY =1, čeka se dok ne postane BUSY = 0. Treba obratiti pažnju na liniju 124 u kojoj je trenutno naredba NOP. Ukoliko u štampaču nema automatskog generisanja LF (novi red) posle CR (kraj reda), ovde treba da stoji LDA #10. Program je smešten u oblast slobodnih lokacija $2A7 – $2FF.

Pri radu u bejziku inicijalizacija se vrši sa SYS 723, a zatim sledi OPEN 4,4:CMD 4 posle čega PRINT i LIST šalju podatke na štampač umesto na ekran. Na kraju dolazi PRINT#4 (UNLISTEN) i CLOSE 4.

```
10 REM *** CENTRONIKS ***
20 REM ******************
30 :
100 SYS 8*4096
101 .OPT OO,P
102 :OFLTO - $9A
103 :CHROUT - $F1CA
104 :CR - 13:LF - 10
105 :PRA - $0000
106 :PRB - PRA+1
107 :OORA - PRA+2
108 :ODRB - PRA +3
109 :ICR - $D000
110 :I8SOUT - $326
111 *- $2A7
112 ;
113 ; NOVA RUTINA ZA ISPISIVANJE KARAKTERA
114 ; ------------------------------------
115 :NEWCHR PHA
116 :        LOA DFLTD
117 :        CMP #4        ; DA LI JE IZLAZNI UREOJAJ #4
118 :        BEQ SKIP
119 :        JMP CHROUT+3; AKO NIJE,PDVRATAK U KERNAL CHROUT RUTINU
```

```
:SKIP   PLA
:       CMP #CR      ; DA LI JE KRAJ REDA
:       BNE CENT     ; AKO NIJE, POSLATI BAJT NA IZLAZ
:       JSR CENT     ; POSLATI <CR> NA IZLAZ
:       NOP
;
; RUTINA ZA SLANJE BAJTA NA IZLAZ
; -------------------------------
:CENT   STA PRB      ; BAJT U REGISTAR "B" CIA#2 (IZLAZ)
:       LDA PRA      ; GENERISANJE STROBE IMPULSA
:       AND #$FB
:       STA PRA      ; STROBE=0
:       ORA #$04
:       STA PRA      ; STROBE=1
;
; AKO JE BUSY=1, CEKAJ
; --------------------
:WAIT   LDA ICR
:       AND #$10
:       BEQ WAIT
:       CLC
:       RTS
;
; CIA#2 SE INICIJALIZUJE DA RADI PO CENTRONIKS PROTOKOLU
; -------------------------------------------------------
:INIT   SEI
:       LDA #$FF
:       STA DDRB     ; PRB IZLAZNI
:       LDA DDRA
:       ORA #$04
:       STA DDRA     ; PA2 IZ PRA IZLAZNI (STROBE)
:       LDA PRA
:       ORA #$04
:       STA PRA      ; STROBE=1
:       LDA #10
:       STA ICR      ; ICR MASK BIT#4=1. ZABRANA PREKIDA PREKO FLAG
:       LDA ICR
;
; IZMENA VEKTORA CHROUT RUTINE
; ----------------------------
:       LDA #<NEWCHR
:       STA IBSOUT
:       LDA #>NEWCHR
:       STA IBSOUT+1
:       CLI
:       RTS
.END
```

12.2 RS 232 INTERFEJS

RS 232 je jedan od standarda kojima se određuje način prenosa podataka. Koristi se pri komunikaciji računara sa drugim računarom, terminalom, štampačem itd. Radi se o dvosmernom serijskom asinhronom prenosu.

U Komodorovom Kernalu nalaze se rutine za softversku emulaciju (imitiranje) specijalizovanog integrisanog kola za serijski prenos — UART-a 6551. Njegovi pseudo registri nalaze se u oblasti sistemskih promenljivih $293 – $297 (659 – 663). Hardverski izvodi se nalaze na korisničkom priključku i imaju sledeće značenje.

izvod	naziv	opis
A	GND	zaštitna masa
B	RX	prijemni podaci
C	RX	prijemni podaci
D	RTS	zahtev modemu za slanje podataka
E	DTR	računar spreman za rad sa modemom
F	RI	u modemu je detektovan poziv
H	DCD	prisutan je nosilac
K	CTS	modem spreman za slanje po prijemu RTS
L	DSR	podaci u računaru spremni za slanje
M	TX	predajni podaci
N	GND	signalna masa

U slučaju da se ne radi sa modemskim kontrolnim linijama, koriste se linije RX, TX i signalna masa. Nivoi signala su TTL (0 i 5V po V24 protokolu). Kernal rutine koje upravljaju RS 232 vezom, pod kontrolom su NMI prekida koje generiše CIA 2. Podaci se smeštaju privremeno u prjemni ili predajni bafer (oba su dužine po 256 bajtova) bez obzira na program koji se trenutno izvršava. Zbog toga prenos podataka preko RS 232 može da bude konkurentan sa izvršavanjem nekog programa bez međusobnog uticaja. Čitanje i upisivanje u bafere može se obaviti iz bejzika i iz mašinskog jezika. U bejziku se RS 232 kanal otvara na sledeći način:

OPEN d,2,a,"(ctrl.reg) + (com.reg)
d — logički broj datoteke
a — sekundarna adresa
(ctrl. reg) — sadržaj kontrolnog registra. Određivanje broja bita u reči i brzine prenosa.
(com. reg) — sadržaj komandnog registra. Određuje način prenosa. Nije obavezno navesti. Za detalje pogledati lokacije $293 — $297 u poglavlju 8. Organizacija memorije i upotreba ROM rutina.

Čitanje podataka iz RS 232 bafera vrši se naredbom GET#. Upotreba INPUT # se ne preporučuje jer, ukoliko se bafer isprazni, a novi podaci ne pristižu sa ulaza, računar može da čeka u beskonačnoj petlji. Iz istog razloga ne treba koristiti ni Kernal CHRIN rutinu.

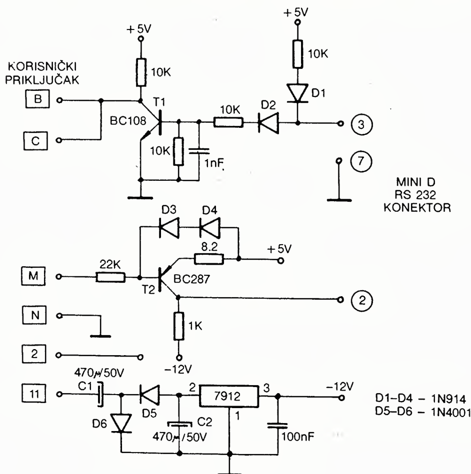
Podaci se šalju preko RS 232 naredbama CMD ili PRINT # dok se kanal zatvara naredbom CLOSE d.

Kod programiranja u mašinskom jeziku koriste se sledeće Kernal rutine:

SETLFS, SETNAM i OPEN	za otvaranje kanala
CHKIN, GETIN	za čitanje bafera
CHKOUT, CHROUT	za upisivanje u bafer
CLOSE	za zatvaranje kanala
READST	za ispitivanje statusa posle izvršavanja pojedinih rutina

Sledeći program omogućuje Komodoru da radi kao terminal. Brzina razmene podataka je 300 Baud-a full duplex tako da može da se koristi za komunikaciju preko V21 modema. Prvi deo programa vrši pretvaranje PETASCII koda u standardni ASCII kod. Drugi deo je prijemni, a treći predajni.

Ukoliko se radi po RS 232C standardu (uobičajeno kod komunikacije sa terminalom ili štampačem), nivou nule odgovara napon od +3 do +15V, a nivou jedinice napon od −3



Sl. 12. 2. Šema RS 232C interfejsa

do −15V. Pomoću dodataka sa slike 12.2 zadovoljava se i ovaj zahtev. Napon od − 12V dobija se od naizmeničnog napona 9V dostupnog na korisničkom priključku. On se udvostručuje pomoću D5, D6, C1 i C2, a zatim stabiliše kolom 7912. Tranzistori T1 i T2 obezbeđuju potrebnu inverziju i prilagođenje signala.

```
10 REM ******************
20 REM *  RS 232 TERMINAL *
30 REM ******************
40 :
100 OPEN 5,2,3,CHR$(6):REM 300 BAUD-A
101 :
105 REM PRETVARANJE PETASCII U ASCII
106 REM ----------------------------
107 :
110 DIM F%(255),T%(255)
200 FOR J=32 TO 64:T%(J)=J:NEXT
```

```
210 T%(13)=13:T%(20)=8
211 RV=18:REM RVS ON
212 CT=0:REM POCETNA VREDNOST BROJACA INTERVALA TREPTANJA KURSORA
220 FOR J=65 TO 90:K=J+32:T%(J)=K:NEXT
230 FOR J=91 TO 95:T%(J)=J:NEXT
240 FOR J=193 TO 218:K=J-128:T%(J)=K:NEXT
250 T%(146)=16:T%(133)=16
260 FOR J=0 TO 255
270 K=T%(J)
280 IF K<>0 THEN F%(K)=J:F%(K+128)=J
290 NEXT
300 PRINT " "CHR$(147)
301 :
302 REM PRIJEM KARAKTERA SA RS 232
303 REM -------------------------
304 :
310 GET#5,A$
320 IF A$=""OR ST<>0 THEN 360
330 PRINT " "CHR$(157);CHR$(F%(ASC(A$)));
340 IF F%(ASC(A$))=34 THEN POKE212,0:REM AKO JE KARAKTER ZNAK NAVODA
345 REM                              PONISTITI "QUOTE" NACIN RADA
350 GOTO 310
351 :
352 REM SLANJE KARAKTERA PREKO RS 232
353 REM ----------------------------
354 :
360 PRINT CHR$(RV)" "CHR$(157);CHR$(146);:GET A$
370 IF A$<>""THENPRINT#5,CHR$(T%(ASC(A$)));
380 CT=CT+1:REM UVECAJ SADRZAJ BROJACA
390 IF CT=8 THEN CT=0:RV=164-RV:REM AKO JE BROJAC DOSAO DO KRAJA ONDA RVS OFF
410 GOTO 310
420 END
```

12.3 MODEM

Modem (modulator — demodulator) je uređaj koji omogućuje razmenu podataka između dva računara preko standardne telefonske linije. Podaci se šalju i primaju serijski bit po bit. Pri slanju se frekventno moduliše nosilac audio frekvencije tako da se dobija jedna frekvencija za nulu, a druga za jedinicu (engl. FSK — frequency shift keying). Obe frekvencije moraju da se nađu unutar frekventnog opsega telefonskog kanala (300Hz — 3400Hz). Za amaterske potrebe najinteresantniji je prenos po CCITT standardu brzinom od 300Bauda u full duplex-u (V21 protokol). Zbog toga se modem priključuje direktno na aRS 232 koji radi sa TTL nivoima (V24 protokol) i podešen je na navedeni standard. Fizički postoji razlika u predajnim frekvencijama uređaja koji započinje komunikaciju i onog koji odgovara.

funkcija	vrednost bita	frekvencija po CCITT (Hz)
kanal 1	0	1180
(započinje)	1	980
kanal 2	0	1850
(odgovara)	1	1650

Pošto direktno povezivanje na telefonsku liniju nije dozvoljeno bez odgovarajućeg atesta, koristi se tzv. akustička sprega (engl. acoustic coupling). Naime, signal se akustički šalje preko mikrofona, a prima sa slušalice telefonskog aparata.

Sl. 12. 3. Šema modema

Akustički signal se iz telefonske slušalice kristalnim mikrofonom pretvara u električni. On se zatim pojačava oko 100 puta i vodi na digitalni filter MF10 (proizvodi ga National Semiconductor). To je dvostruki propusnik opsega čija centralna frekvencija zavisi od frekvencije takta na ulazima A i B. Ovu frekvenciju generiše VCO 74LS625 koji je podešen trimerima od 1K na 108KHz (kanal 1) i 175KHz (kanal 2). Filtrirani signal se vodi na komparator LM 339 koji formira četvrtasti talasni oblik, a zatim na RX ulaz modemskog integrisanog kola MC 14412. Demodulisani signal se preko izlaza RX DATA i pojačavača vodi na RS 232 priključak Komodora.

Pri slanju podataka signal se prvo vodi na bafer BC 238, a zatim na ulaz modema. Modulisani signal se pojačava Darlington parom BC 286 – BD 131 pa se preko zvučnika dobija akustički signal koji se dalje šalje u sprezi sa telefonskim mikrofonom.

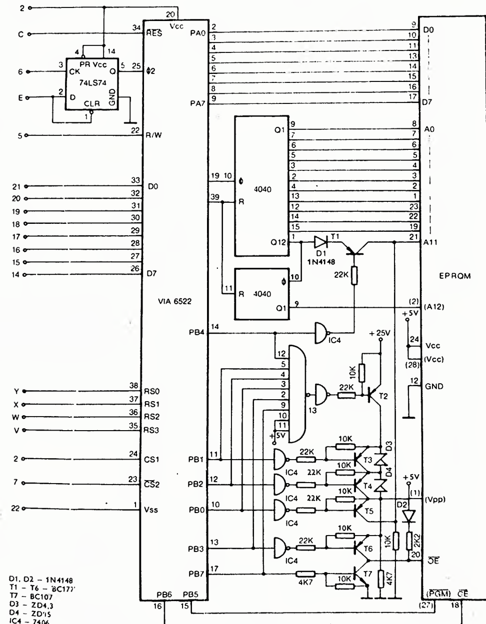
Kada je preklopnik P2 u položaju TEST, podaci koje šalje računar prolaze kroz modem i vraćaju se nazad u računar tako da može da se proveri njihova ispravnost.

12.4 EPROM PROGRAMATOR

Na slici 12.4 je prikazan jednostavan EPROM programator za programiranje EPROM-a tipa 2716, 2732 (2532) i 2764.

Perferna jedinica za paralelni i serijski prenos podataka, VIA 6522 je slična već postojećim u Komodoru (CIA 1 i CIA 2). Pošto je u ovom slučaju smešten u Komodorov I/0 1 adresni prostor ($DE00 – $DEFF) adrese pojedinih registara su:

registar	adresa	funkcija
DRB	DE00	ulazno/izlazni registar B
DRA	DE01	ulazno/izlazni registar A + kontrola
DDRB	DF02	registar smera za B
DDRA	DE03	registar smera za A
T1CL	DE04	registar za čitanje tajmera (niži bajt)
T1CH	DE05	registar za čitanje tajmera (viši bajt)
T1LL	DE06	tajmer 1 leč registar (niži bajt)



Sl. 12. 4. Šema EPROM programatora

T1LH	DE07	tajmer 1 leč registar (viši bajt)
T2CL	DE08	tajmer 2 niži bajt
T2CH	DE09	tajmer 2 viši bajt
SR	DE0A	serijski ulazno/izlazni registar
ACR	DE0B	pomoćni kontroini registar
PCR	DE0C	periferni kontrolni egistar
IFR	DE0D	registar prekida
IER	DE0E	registar maski prekida
DRA	DE0F	U/I registar A bez kontrole

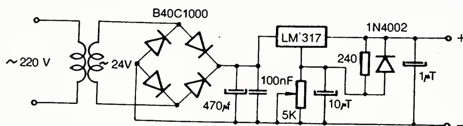
VIA 6522 je direktno kompatibilna sa magistralama mikroprocesora 6510 pa se direktno priključuje na priključak za proširenja. Preko nje se vrši kompletno softversko upravljanje programiranjem EPROM-a. Podaci se prenose preko kapije A (PA0 do PA7) dok se adresa prenosi serijski preko CB2 (registar PCR) a zatim se pretvara u paralelni oblik u brojačima 4040. Kapija B se koristi za biranje odgovarajućeg rasporeda priključaka kao i napona programiranja za razne tipove EPROM-a. Oznake za 2716 i 2732 date su standardno dok su oznake za 2764 date u zagradama. Flip flop 74LS74 služi za korigovanje odnosa signal – pauza takta Φ2. Impuls od 50 ± 5ms se dobija korišćenjem tajmera 1. Pošto je sistemski takt 985 KHz, potrebno je 49250 ciklusa da se dobije tačna vrednost.

```
10 SYS 8*4096
20 .OPT 00
30 *- $2CD
40 DRB - $DEOD
50 T1CH - $DE05
70 IFR - $DEOD
75 ;
80 ; GENERISANJE IMPULSA OD 50 MS
90 ; ----------------------------
100 :         SEI
110 :         LDA #$47
120 :         STA ORB
130 :         LDA #$CD
140 :         STA T1CH
150 :WAIT     LDA IFR
160 :         AND #$40
170 :         BEQ WAIT
180 :         LDA #$D7
190 :         STA DRB
200 :         CLI
210 :         RTS
220 .ENO
10 REM *** EPROM PROGRAMATOR ***
20 REM *************************
30 :
40 REM POZAOINA CRNA, SLOVA PLAVA
100 POKE S3280,0:POKE S3281,0:PRINT CHR$(154)
110 PRINT CHR$(14);CHR$(8):REM MALA SLOVA
120 IF PEEK (704)-120 THEN 140
130 GOSUB 1220:REM UCITATI MASINSKI PROGRAM
135 :
140 REM INICIJALIZACIJA
145 REM ---------------
146 :
160 DB-S6B32:REM DRB
170 DA-DB+1: REM DRA
180 RB-DB+2: REM DDRB
```

```
190 RA=DB+3: REM DDRA
200 TH=DB+5: REM TIMER1 HI
210 TL=DB+6: REM TIMER1 LO
220 AC=DB+11:REM ACR
230 PC=DB+12:REM PCR
240 MS=704:M1=MS+2:M2=MS+19
260 POKE TL,98: REM POSTAVLJANJE NIZEG BAJTA TAJMERA 1
270 POKE RB,255:POKE AC,0
275 :
280 REM MENUE
285 REM -----
286 :
290 PRINT"{CLR}{C/DN}{C/DN}      TIP EPROMA{C/DN}{C/DN}"
300 PRINT"{C/DN}        1 - 2716"
310 PRINT"{C/DN}        2 - 2732"
320 PRINT"{C/DN}        3 - 2732A"
330 PRINT"{C/DN}        4 - 2764"
340 POKE 198,0:WAIT 198,1
350 GET A
355 ON A GOTO 360,370,380,390
358 IF A=0 OR A>4 THEN 340
360 Z=2048:L=129:PN=7 :PI=71:SB=65:GOTO 410
370 Z=4096:L=144:PN=94:PI=30:SB=80:GOTO 410
380 Z=4096:L=144:PN=92:PI=28:SB=80:GOTO 410
390 Z=8192:L=176:PN=52:PI=20:SB=112
410 POKE DB,SB
420 PRINT"{CLR}{C/DN}{C/DN}STAVI EPROM I PRITISNI TASTER"
430 POKE 198,0:WAIT 198,1
440 PRINT"{CLR}{C/DN}{C/DN}IZABERI OPERACIJU{C/DN}{C/DN}"
450 PRINT"{C/DN} C - CITANJE"
460 PRINT"{C/DN} T - TESTIRANJE"
470 PRINT"{C/DN} V - VERIFIKOVANJE"
480 PRINT"{C/DN} P - PROGRAMIRANJE"
490 PRINT"{C/DN} K - KRAJ"
500 POKE 198,0:WAIT 198,1
510 GET A$:IF A$<>"K" THEN 550
520 POKE DB,SB
530 PRINT"{C/DN}{C/DN}IZVADI EPROM I PRITISNI TASTER"
540 POKE 198,0:WAIT 198,1:END
550 IF A$="C" THEN GOSUB 640:GOTO 610
560 IF A$="T" THEN GOSUB 720:GOTO 610
570 IF A$="V" THEN GOSUB 800:GOTO 610
580 IF A$<>"P"THEN 500
590 GOSUB 960
610 POKE DB,SB
620 PRINT" - PRITISNI TASTER"
630 POKE 198,0:WAIT 198,1:GOTO 440
635 :
640 REM CITANJE EPROMA
645 REM --------------
646 :
650 PRINT"{CLR}{C/DN}{C/DN} POCETNA ADRESA RAM BAFERA (HEX)"
660 GOSUB 1150:SA=DE
670 PRINT"{C/DN}{C/DN}      CITANJE U TOKU"
680 POKE PC,223:POKE PC,221:POKE DB,1
690 FOR I=0 TO Z-1
700 POKE SA+I,PEEK(DA):POKE PC,253:POKE PC,221:NEXT
710 PRINT"{C/DN}{C/DN}       KRAJ CITANJA";:RETURN
713 :
714 REM TESTIRANJE EPROMA
715 REM -----------------
720 :
730 FE=0:PRINT"{CLR}{C/DN}{C/DN}      LOKACIJE KOJE NISU OBRISANE"
735 PRINT"      ----------------------------"
736 PRINT"{C/DN}";TAB(10)"ADRESA";TAB(20)"PODATAK{C/DN}"
740 POKE PC,223:POKE PC,221:POKE DB,1:POKE RA,0
```

```
750 FOR I=0 TO Z-1
760 D=PEEK(DA)
770 IF D<>255 THEN PRINT TAB(10)I;TAB(20);D:FE=1
780 POKE PC,253:POKE PC,221:NEXT:IF FE=0 THEN PRINT "{C/DN}      OK";
790 RETURN
793 :
795 REM VERIFIKOVANJE EPROMA
796 REM --------------------
800 :
810 PRINT"{CLR}{C/DN}{C/DN} POCETNA ADRESA RAMA (HEX)"
820 GOSUB 1150:SA=DE
830 PRINT"{C/DN}{C/DN} POCETNA ADRESA EPROMA (HEX)"
840 GOSUB 1150 :EA=DE
850 Z=Z-EA:PRINT"{C/DN}{C/DN} BROJ BAJTOVA{C/DN}":INPUT BY
860 IF BY>Z THEN PRINT "{CLR}{C/DN} SUVISE VELIKI BROJ!"
870 IFBY>ZTHEN PRINT"{C/DN} MAX."Z:PRINT" OD POC.ADRESE"EA:Z=Z+EA:GOTO 830
880 POKE PC,223:POKE PC,221
885 PRINT"{CLR}{C/DN}{C/DN}       VERIFIKOVANJE U TOKU"
890 IF EA>0 THENFOR I=1 TO EA:POKE PC,253:POKEPC,221:NEXT
900 PRINT:POKE DB,1:POKE RA,0
910 FE=0:FOR I=0 TO BY-1
920 D=PEEK(DA)
930 IF D<>PEEK(SA+I) THEN PRINT"     GRESKA U";I+EA:FE=1
940 POKE PC,253:POKE PC,221:NEXT:IF FE=0 THEN PRINT:PRINT"         OK";
950 RETURN
955 :
956 REM PROGRAMIRANJE EPROMA
957 REM --------------------
960 :
970 PRINT"{CLR}{C/DN}{C/DN} POCETNA ADRESA RAM PODATAKA (HEX)"
980 GOSUB 1150:SA=DE
990 PRINT"{C/DN}{C/DN} POCETNA ADRESA U EPROMU (HEX)"
1000 GOSUB 1150 :EA=DE
1010 Z=Z-EA
1020 PRINT"{C/DN}{C/DN}BROJ BAJTOVA{C/DN}"
1030 INPUT BY
1040 IF BY>Z THEN PRINT "{CLR}{C/DN} SUVISE VELIKI BROJ!"
1050 IFBY>ZTHEN PRINT"{C/DN} MAX."Z:PRINT" OD POC.ADRESE"EA:Z=Z+EA:GOTO 990
1060 PRINT"{CLR}{C/DN}{C/DN}       PROGRAMIRANJE U TOKU"
1070 POKE PC,223:POKE PC,221
1080 IF EA<>0 THEN FOR I=1 TO EA:POKE PC,253:POKE PC,221:NEXT
1090 POKE RA,255:POKE M1,PI:POKEM2,PN:POKE DB,PN
1100 FOR I=0 TO BY-1
1110 D=PEEK(SA+I)
1120 POKE DA,D:SYS MS
1130 POKE PC,253:POKE PC,221:NEXT:PRINT:PRINT"{C/UP}{C/UP}      KRAJ";:RETURN
1135 :
1136 REM PRETVARANJE HEX U DEC
1140 REM ---------------------
1145 :
1150 INPUT"{C/DN}";H$:IFLEN(H$)=0ORLEN(H$)>4THENPRINT"{C/DN} GRESKA":GOTO1150
1155 DE=0
1165 FORWP=1 TO LEN(H$)
1166 FF$=MID$(H$,LEN(H$)+1-WP,1)
1167 IF ASC(FF$)>57 THENXX=ASC(FF$)-55:GOTO 1169
1168 XX=VAL(FF$)
1169 DE=DE+XX*(16^(WP-1))
1170 NEXT
1171 PRINT" DECIMALNO";DE
1172 RETURN
1205 :
1206 REM LOADER MASINSKOG PROGRAMA
1210 REM ------------------------
1215 :
1220 MS=704:FOR I=MS TO MS+24:READ DC:A=A+DC:POKE I,DC:NEXT
1230 IF A<>3177 THEN PRINT "GRESKA U DATA LINIJAMA":END
1240 DATA 120,169,71,141,0,222,169,192,141,5,222,173,13,222,41,64,240,249
1250 DATA 169,7,141,0,222,88,96
1260 RETURN
```

Napon za programiranje (Vpp) može se dobiti pomoću kola sa sl. 12.5.



Sl. 12. 5. Šema ispravljača i regulatora za dobijanje napona od 25V

Trimer potenciometrom od 5Koma izlazni napon se podešava na +25V.

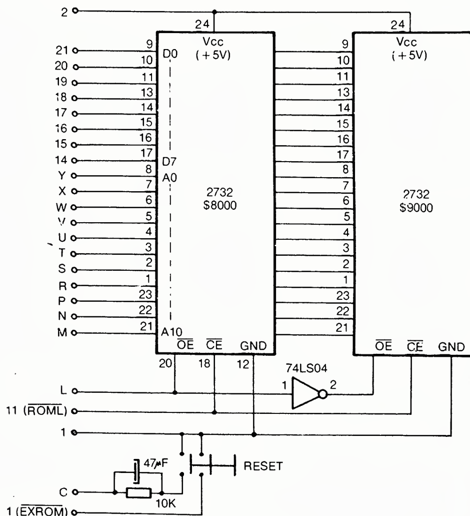
12.5 **ROM MODUL**

Na priključak za proširenje Komodora može se priključiti i EPROM modul poznat kao kartridž. Program u njemu se može izvršiti odmah po uključenju računara (videti Kernal rutinu AOINT) ili se može pozivati iz nekog drugog programa.

Iz blok šeme organizacije memorije može se videti da postoje tri memorijska segmenta gde se može nalaziti EPROM modul. Najčešće se koristi oblast $8000 – $9FFF gde se ROM aktivira postavljanjem linije $\overline{\text{EXROM}}$ na nulu.

Ukoliko postoji autostart zaglavlje, program u ROM-u će početi da se izvršava odmah po uključenju, odnosno, resetovanju računara (videti Kernal rutinu RESET). Naravno, moguće su i druge konfiguracije priključivanja spoljašnjeg EPROM-a zavisno od stanja na linijama $\overline{\text{EXROM}}$, $\overline{\text{GAME}}$, $\overline{\text{LORAM}}$, $\overline{\text{HIRAM}}$ kao i od izbora linije za selekciju ROM-a ($\overline{\text{ROMH}}$ ili ROML).

priključeni	sigmenti	$\overline{\text{EXROM}}$	$\overline{\text{GAME}}$	$\overline{\text{LORAM}}$	$\overline{\text{HIRAM}}$	linija za selekciju
$8000 – $9FFF $A000 – $BFFF $E000 – $FFFF	EPROM bejzik ROM Kernal ROM	0	1	1	1	$\overline{\text{ROML}}$
$8000 – $9FFF $A000 – $BFFF $E000 – $FFFF	RAM RAM Kernal ROM	0	1	0	1	–
$8000 – $9FFF $A000 – $BFFF $E000 – $FFFF	EPROM 1 – EPROM 2	1	0	X	X	$\overline{\text{ROML}}$ $\overline{\text{ROMH}}$
$8000 – $BFFF $E000 – $FFFF	16K EPROM Kernal ROM	0	0	1	1	$\overline{\text{ROML}}$
$8000 – $9FFF $A000 – $BFFF $E000 – $FFFF	RAM EPROM Kernal ROM	0	0	0	1	$\overline{\text{ROMH}}$



Sl. 12. 6. Šema ROM modula

Dodatak

SKRAĆENI NAZIV PISANJA REZERVISANIH REČI U BEJZIKU

naredba	skraćenica	ekran
ABS	A SHIFT B	A
AND	A SHIFT N	A
ASC	A SHIFT S	A ♥
ATN	A SHIFT T	A
CHR$	C SHIFT H	C
CLOSE	CL SHIFT O	CL
CLR	C SHIFT L	C
CMD	C SHIFT M	C
CONT	C SHIFT O	C
DATA	D SHIFT A	D ♠
DEF	D SHIFT E	D
DIM	D SHIFT I	D
END	E SHIFT N	E
EXP	E SHIFT X	E ♣
PEEK	P SHIFT E	P
POKE	P SHIFT O	P
PRINT	?	?
PRINT#	P SHIFT R	P
READ	R SHIFT E	R
RESTORE	RE SHIFT S	RE ♥
RETURN	RE SHIFT T	RE
RIGHT$	R SHIFT I	R
RND	R SHIFT N	R
RUN	R SHIFT U	R
SAVE	S SHIFT A	S ♠
SGN	S SHIFT G	S
SIN	S SHIFT I	S
FOR	F SHIFT O	F
FRE	F SHIFT R	F
GET	G SHIFT E	G
GOSUB	GO SHIFT S	GO ♥
GOTO	G SHIFT O	G
INPUT#	I SHIFT N	I
LET	L SHIFT E	L
LEFT$	LE SHIFT F	LE
LIST	L SHIFT I	L
LOAD	L SHIFT O	L
MID$	M SHIFT I	M
NEXT	N SHIFT E	N
NOT	N SHIFT O	N
OPEN	O SHIFT P	O
SPC(	S SHIFT P	S
SQR	S SHIFT Q	S ●
STEP	ST SHIFT E	ST
STOP	S SHIFT T	S
STR$	ST SHIFT R	ST
SYS	S SHIFT Y	S
TAB	T SHIFT A	T ♠
THEN	T SHIFT H	T
USR	U SHIFT S	U ♥
VAL	V SHIFT A	V ♠
VERIFY	V SHIFT E	V
WAIT	W SHIFT A	W ♠

TABELA EKRANSKIH KODOVA

skup 1	skup 2	kôd	skup 1	skup 2	kôd	skup 1	skup 2	kôd
@		0	↑		30	<		60
A	a	1	←		31	=		61
B	b	2	SPACE		32	>		62
C	c	3	!		33	?		63
D	d	4	"		34			64
E	e	5	#		35	♠	A	65
F	f	6	$		36		B	66
G	g	7	%		37		C	67
H	h	8	&		38		D	68
I	i	9	'		39		E	69
J	j	10	(		40		F	70
K	k	11	)		41		G	71
L	l	12	*		42		H	72
M	m	13	+		43		I	73
N	n	14	,		44		J	74
O	o	15	–		45		K	75
P	p	16	.		46		L	76
Q	q	17	/		47		M	77
R	r	18	0		48		N	78
S	s	19	1		49		O	79
T	t	20	2		50		P	80
U	u	21	3		51	●	Q	81
V	v	22	4		52		R	82
W	w	23	5		53	♥	S	83
X	x	24	6		54		T	84
Y	y	25	7		55		U	85
Z	z	26	8		56	╳	V	86
[		27	9		57	○	W	87
£		28	:		58	♣	X	88
]		29	;		59		Y	89

skup 1	skup 2	kôd	skup 1	skup 2	kôd	skup 1	skup 2	kôd
	Z	90			103			116
		91			104			117
		92			105			118
		93			106			119
		94			107			120
		95			108			121
SPACE		96			109			122
		97			110			123
		98			111			124
		99			112			125
		100			113			126
		101			114			127
		102			115			

Kodovi od 128 do 255 daju iste kraktere kao i kodovi od 0 do 127, ali ispisane inverzno.

TABELA PETASCII KODOVA

karakter	kôd (dec)	karakter	kôd (dec)	karakter	kôd (dec)	karakter	kôd (dec)
	0		12		24	$	36
	1	RETURN	13		25	%	37
	2	uključi skup 2	14		26	&	38
	3		15		27	'	39
	4		16	RED	28	(	40
WHT	5	CRSR	17	CRSR	29	)	41
	6	RVS ON	18	GRN	30	*	42
	7	CLR HOME	19	BLU	31	+	43
onemogući SHIFT C=	8	INST DEL	20	SPACE	32	,	44
omogući SHIFT C=	9		21	!	33	−	45
	10		22	"	34	.	46
	11		23	#	35	/	47

karakter	kôd (dec)	karakter	kôd (dec)	karakter	kôd (dec)	karakter	kôd (dec)
0	48	O	79		110	SHIFT RETURN	141
1	49	P	80		111	uključi sklop 1	142
2	50	Q	81		112		143
3	51	R	82		113	BLK	144
4	52	S	83		114	CRSR	145
5	53	T	84	♥	115	RVS OFF	146
6	54	U	85		116	SHIFT CLR HOME	147
7	55	V	86		117	INST DEL	148
8	56	W	87		118	braon	149
9	57	X	88		119	sv. crvena	150
:	58	Y	89	♣	120	siva 1	151
;	59	Z	90		121	siva 2	152
<	60	[	91	♦	122	sv. zelena	153
=	61	£	92		123	sv. plava	154
>	62	]	93		124	siva 3	155
?	63	↑	94		125	PUR	156
@	64	←	95		126	CRSR	157
A	65		96		127	YEL	158
B	66	♠	97		128	CYN	159
C	67		98	narandžasta	129	SPACE	160
D	68		99		130		161
E	69		100		131		162
F	70		101		132		163
G	71		102	f1	133		164
H	72		103	f3	134		165
I	73		104	f5	135		166
J	74		105	f7	136		167
K	75		106	f2	137		168
L	76		107	f4	138		169
M	77		108	f6	139		170
N	78		109	f8	140		171

karakter	kôd (dec)	karakter	kôd (dec)	karakter	kôd (dec)	karakter	kôd (dec)
	172		177		182		187
	173		178		183		188
	174		179		184		189
	175		180		185		190
	176		181		186		191

Literatura

1. "Computer's First Book of Commodore 64", Computer's publications, Greensboro: 1983., ISBN 0-942386-20-S
2. "Computer's First Book of Commodore 64 Sound and Graphics", Computer's publications, Greensboro: 1983., ISBN 0-942386-21-3
3. S. Lenon; "Mapping the Commodore 64", Computer's publications, Greensboro: 1984., ISBN-942386-23-X
4. "64 Intern", Data Becker, Duseldorf: 1984., ISBN 3-89011-000-2
5. Z. Salčić; "Mikroprocesorski sistemi", Svjetlost, Sarajevo: 1982
6. D. F. Stout; "Microprocessor Applications Handbook", McGraw Hill, New York: 1982., ISBN 0-07-61798-8
7. J. B. Peatman; "Microcomputer-Based Design", McGraw Hill, New York: 1977., ISBN 0-07--049138-0
8. A. Ralston; "Encyclopedia of Computer Science", Van Nostrand Reinhold, New York: 1976., ISBN 0-442-80321-4
9. H. Lorin, H. Ditel; "Operating Systems", Adison-Wesley, Reading: 1981., ISBN 0-201-14464-6
10. P. Calingaret; "Assemblers, Compilers and Program Translation", Computer Science Press, Potomac: 1979., ISBN 0-914894-23-4
11. "Programmer's Reference Guide", Commodore Business Machines Inc, West Chester PA 19380, USA: 19a2., ISBN 0-672-220S6-3
12. „VIC-1S41 Single drive floppy disk user's manual", Commodore Business Machines, Inc.: 1981
13. "Simon's Basic 64, Modul-Version uber 100 zusatzliche Basic Befehle", Commodore GmbH, Frankfurt: 1984
14. Plenge, Szczepanowski; "Das Trainingsbuch zum Simons's Basic", Data Becker Buch, Deutchland: 1984
15. D. Davis; "Machine Language for the Absoloute Beginner", Melbourne House, United Kingdom: 1984., ISBN 0-86161-14S-4
16. "Commodore 64 priručnik za upotrebu", Konim, Mladinska knjiga, Ljubljana: 198S
17. L. Popović, M. Popović; "Commodore I/O", Beograd biro, Beograd: 198S
18. J. Dujmović; "Mikroprocesorski sistemi", Beleške sa predavanja na elektrotehničkom fakultetu, Beograd: 1981
19. S. Muftić; "Osnovni elementi kompjuterskih sistema", Svjetlost, Sarajevo: 1983
20. S. Alagić; "Principi programiranja", Svjetlost, Sarajevo: 1982
21. J. Žitnik, I. Kononeko; "Tehnika programiranja", Iskra, Iskra Delta, Ljubljana: 198S
22. "Moj Mikro", časopisi, ČGP Delo, Ljubljana: 198S
23. "Računari", časopisi, BIGZ, Beograd: 198S
24. "Svet Kompjutera", časopisi, NO "Politika", Beograd: 198S
25. "64'er das magazin fur computer-fans", Markt & Technik, Haar bei München: 1985
26. "Electronic Engineering, Morgan Grampian House, 30 Calderwood Street, Wooldwich, London SE18 6QH, ISSN 0013-4902
27. "Mini Micro System", Cahners Publishing Company, Division of Reed Holdings, Inc., 221 Columbus Avenue, Boston, MA 02116, ISSN 0364-9342
28. "Byte", Byte Publication Inc., 70 Main St. Peterborough NH 034S8, ISSN 0360-S280
29. J. Millman, C. C. Halkias; "Integrated Electronics", McGraw Hill Kogakusha, Ltd., Tokyo: 1972
30. H. Gunter Steidle "Tranzistorske priručne tabele", Partizanska knjiga, Znanstveni tisk, Ljubljana: 1984
31. "Linear Integrated Circuits and MOS/FETs databook", R.C.A., SSD 240B-E: 1984
32. "CMOS Integrated Circuits Databook", R.C.A., SSD-250C, 1984
33. "JUS A.FO.001, Obrada informacija - Definicije pojmova", Jugoslovenski zavod za standardizaciju, Beograd: 1979

Mesečna revija »Moj mikro« redovno donosi:

- testove i prezentacije novih kućnih i ličnih računara
- detaljne opise periferne opreme
- hardverske savete
- programe i softverske članke
- tematske priloge
- recenzije domaće i strane literature
- izveštaje sa svetskih sajmova
- igre

Karakteristike revije »Moj mikro«:

- Veliki broj autora iz cele Jugoslavije
- kvalitetna štampa (20 strana u boji na finom papiru)
- prednost imaju originalni članci
- najveća berza malih oglasa u Jugoslaviji
- najjači odjek među komercijalnim oglašivačima
- opširne kontaktne rubrike (pisma, odgovori na stručna pitanja, nagradne igre i zagonetke itd.)

Izdanja Mikro knjige najbrže možete nabaviti
direktno od izdavača:

MIKRO KNJIGA
P. O. Box 75
11090 Rakovica–BEOGRAD

Takođe ih možete nabaviti
i u svim bolje snabdevenim knjižarama širom Jugoslavije.
Neke od njih su:

BEOGRAD:	„Tehnička knjiga", prodavnica 1, Maršala Tita 21
	„Jugoslovenska knjiga", prodavnica 1, Knez Mihajlova 2
	„Prosveta", knjižara 1, Knez Mihajlova 12
NOVI SAD:	„NOLIT 52", Sutjeska 2
NIŠ:	„Gradina", prodavnica 2, Maršala Tita 38
ZAGREB:	„Mladinska knjiga", Trg bratstva i jedinstva 3
	„Mladost", prodavnica 1, Ilica 30
SPLIT:	„Znanstvena knjiga", Trg Preporoda 7
RIJEKA:	„Mladost", Korzo 11
LJUBLJANA:	„Mladinska knjiga", Titova 3
MARIBOR:	„Mladinska knjiga", Partizanska 9
SARAJEVO:	„Veselin Masleša", Maršala Tita 52
BANJA LUKA:	„Kultura", Maršala Tita 67a
SKOPLJE:	„Makedonska knjiga", Trgovački centar bb

Index

A
ABS 47
ADC 155
AND 49,157
ANGL 122
ARC 122
ASC 54
ASL 159
AT 108
ATN 48
AUTO 89

B
BCC 161
BCKGNDS 111
BCS 161
BEQ 161
BFLASH 110
BFLASH 0 111
BIT 158
BLOCK 120
BMI 161
BNE 161
BPL 161
BRK 163
BVC 161
BVS 161

C
CALL 103
CENTRE 107
CGOTO 103
CHAR 124
CHECK 132
CHR$ 54
CIRCLE 121
CLC 163
CLD 163
CLI 163
CLOSE 70
CLR 36
CMD 72
CMP 157
COLD 94
COLOUR 111
CONT 22,31
COPY 138
COS 47
CPX 157
CPY 157
CSET 117

D
DATA 39
DEC 157
DEF FN 51
DELAY 91
DESIGN 0 127
DESIGN 1 127
DESIGN 2 126
DETECT 131
DEX 157
DEY 157
DIM 44
DIR 136
DISAPA 97
DISK 136
DISPLAY 93
DIV 104
DOWN 115
DRAW 123
DUMP 94
DUP 107

E
ELSE 99
END 30
END LOOP 101
END PROC 102
ENVELOPE 134
EOR 158
EXEC 102
EXIT IF 101
EXOR 105
EXP 46

F
FCHR 112
FCOL 112
FETCH 98
FILL 113
FIND 91
FLASH 110
FN 51
FOR 42
FRAC 104
FRE 56

G
GET 37
GET# 70
GLOBAL 103
GOSUB 40
GOTO 39

H
HI COL 118
HIRES 115
HRDCPY 138

I
IF 40
IF 99
INC 156
INKEY 98
INPUT 36
INPUT# 71
INSERT 105
INST 106
INT 46
INV 113
INX 157
INY 157

J
JMP 162
JOY 139
JSR 162

K
KEY 92

L
LDA 153
LDX 153
LDY 153
LEFT 114
LEFT$ 53
LEN 54
LIN 109

LINE 120
LIST 20
LIST 29
LOAD 63
LOCAL 103
LOG 46
LOOP 101
LOW COL 118
LSR 159

M
MEM 125
MERGE 90
MID$ 53
MMOB 130
MOB OFF 130
MOB SET 130
MOD 104
MOVE 113
MULTI 116
MUSIC 134

N
NEW 21,29
NEXT 42
NO ERROR 96
NOP 163
NOT 50
NRM 117

O
OFF 110
OLD 95
ON 42
ON ERROR 95
OPEN 69
OPTION 91
OR 49
ORA 158
OUT 97

P
PAGE 90
PAINT 123
PAUSE 109
PEEK 56
PENX 138
PENY 138
PHA 155
PHP 155
PLA 155
PLACE 106
PLAY 135
PLOT 119
PLP 155
POKE 56
POT 139
PRINT 19,31
PRINT# 71
PROC 101

R
RCOMP 100
READ 38
REC 120
REM 31
RENUMBER 90
REPEAT 100
RESET 98
RESTORE 39
RETRACE 94
RETURN 40
RIGHT 114
RIGHT$ 53
RLOCMOB 131
RND 52
ROL 159
ROR 159
ROT 124
RTI 162
RTS 162
RUN 21,29
RUN/STOP 21

S
SAVE 65
SBC 156
SCRLD 137
SCRSV 137
SEC 163
SECURE 97
SED 163
SEI 163
SGN 47
SIN 47
SQR 46
STA 154
STATUS 73
STEP 42
STOP 30
STR$ 54
STX 154
STY 154
SYS 57

T
TAN 47
TAX 152
TAY 152
TEST 119
TEXT 124
THEN 99
TIME 55
TIME$ 55
TO 42
TRACE 93
TSX 152
TXA 152
TXS 152

U
UNTIL 100
UP 115
USE 108
USR 57

V
VAL 55
VERIFY 66
VOL 132

W
WAIT 57
WAVE 133
↑ 46
π 47
$ 105
% 105
? 23
@ 127,129

Sadržaj

OSNOVNI POJMOVI O RAČUNARIMA 7
1.1 RAČUNAR, HARDVER, SOFTVER 7
1.2 BIT, BAJT, BINARNI BROJ 8
1.3 MIKROPROCESOR, MEMORIJA, PERIFERNA JEDINICA 8
1.4 PROGRAMSKI JEZICI 10
1.5 KORISNIČKI I SISTEMSKI PROGRAMI 11

2 UPOTREBA RAČUNARA KOMODOR 64 12

3 UVOD U RAD SA KOMODOROM 16
3.1 SASTAV MIKRORAČUNARSKOG SISTEMA KOMODORA 16
3.2 PUŠTANJE U RAD 16
3.3 TV EKRAN 17
3.4 RAD SA TASTATUROM 17
3.5 NAČINI RADA 19
3.6 UČITAVANJE PROGRAMA 23

4 BEJZIK 25
4.1 OSNOVNI POJMOVI 25
4.2 NAREDBE I NJIHOVA UPOTREBA 28
4.2.1 Osnovne naredbe 29
4.2.2 Aritmetičke operacije 45
4.2.3 Funkcije 46
4.2.4 Trigonometrijske funkcije 47
4.2.5 Operacije poređenja 48
4.2.6 Logičke operacije 49
4.2.7 Prioriteti funkcija i operacija 50
4.2.8 Rad sa stringovima 52
4.2.9 Ostale naredbe 55
4.2.10 Rad sa kasetofonom i diskom 58
4.2.11 Datoteke 67
4.3 IZVEŠTAJI 73

5 PRINCIPI PROGRAMIRANJA 76
5.1 OSNOVNI POJMOVI 76
5.2 RAZVOJ PROGRAMA 77
Analiza problema 77
Algoritam 78
Dijagram toka 79
Osnovne programske strukture 80
Testiranje programa 83
Dokumentacija 83

5.3 STRUKTUIRANO PROGRAMIRANJE 83
Razvoj programa u koracima preciziranja 84
Procedure 84
Lokalne i globalne promenljive 85
Modularnost i adaptibilnost 85

6 SAJMONS BEJZIK 87
6.1 UVOD 87
6.2 NAREDBE I NJIHOVA UPOTREBA 89
6.2.1 Pomoć u programiranju 89
6.2.2 Pomoć pri pojavi greške 93
6.2.3 Kontrola greške 95
6.2.4 Zaštita programskih linija 97
6.2.5 Unošenje podataka 98
6.2.6 Struktuirano programiranje 99
6.2.7 Rad sa brojevima 104
6.2.8 Rad sa stringovima 105
6.2.9 Ispisivanje rezultata 107
6.2.10 Boje na ekranu 110
6.2.11 Pomeranje sadržaja ekrana 114
6.2.12 Grafika 115
6.2.13 Definisanje novih karaktera 125
6.2.14 Sprajtovi 127
6.2.15 Zvuk 132
6.2.16 Rad sa diskom i kasetofonom 135
6.2.17 Rad sa štampačem 137
6.2.18 Rad sa upravljačkim uređajima 138
6.3 IZVEŠTAJI 139
6.4 PRIMER PROGRAMA U SAJMONS BEJZIKU 140

7 PROGRAMIRANJE NA MAŠINSKOM JEZIKU 142
7.1 OD BEJZIKA KA MAŠINSKOM PROGRAMIRANJU 142
7.2 BROJNI SISTEMI 144
Binarni brojevi 144
Apsolutna binarna forma 145
Binarni brojevi u komplementu dvojke 146
Heksadecimalni brojevi 147
Petobajtna forma 147
7.3 MIKROPROCESOR 6510 147
7.4 NAČINI ADRESIRANJA 150
7.5 NAREDBE MIKROPROCESORA 6510 152
7.6 INDIKATORI STANJA 163
7.7 SPISAK NAREDBI MIKROPROCESORA 6510 165
7.8 PRIMERI PROGRAMIRANJA NA MAŠINSKOM JEZIKU . . . 170

8 ORGANIZACIJA MEMORIJE I UPOTREBA ROM RUTINA 181
8.1 ORGANIZACIJA MEMORIJE 181
8.2 SISTEMSKE PROMENLJIVE 182
8.3 BEJZIK INTERPRETER 198
8.3.1 Bejzik interpreter Komodora 199
8.3.2 Organizacija bejzik interpretera 200
8.4 OPERATIVNI SISTEM 211
8.4.1 Dokumentovane rutine i upotreba 213
8.4.2 Organizacija operativnog sistema 230

9 ZVUK . . . 238
9.1 ELEMENTI SINTESAJZERA . . . 238
9.2 GENERISANJE ZVUKA U KOMODORU . . . 240

10 GRAFIKA . . . 246
10.1 REGISTRI VIC-a . . . 246
10.1.1 Kontrolni registri . . . 247
10.1.2 Registri sprajtova . . . 250
10.1.3 Registri boja . . . 251
10.2 RAD SA KARAKTERIMA . . . 251
10.2.1 Standardni karakteri . . . 251
10.2.2 Višebojni karakteri . . . 254
10.2.3 Višebojna pozadina . . . 254
10.3 RAD U VISOKOJ REZOLUCIJI . . . 255
10.3.1 Standardni način rada . . . 255
10.3.2 Višebojni način rada . . . 260
10.4 RAD SA SPRAJTOVIMA . . . 261
10.4.1 Definisanje sprajtova . . . 261
10.4.2 Uključivanje i pozicioniranje sprajtova . . . 262
10.4.3 Sudari . . . 263
10.5 MEŠOVITI NAČIN RADA . . . 264

11 HARDVER . . . 267
11.1 MIKROPROCESOR . . . 267
11.2 RAM . . . 271
11.3 ROM . . . 274
11.4 VIDEO KONTROLER . . . 276
11.5 AUDIO KONTROLER . . . 286
11.6 PERIFERNE JEDINICE I PRIKLJUČCI . . . 288
11.6.1 RF modulator . . . 288
11.6.2 Priključak za kasetofon . . . 289
11.6.3 Audio-video priključak . . . 290
11.6.4 Kompleksni interfejs adapter (CIA) . . . 290
11.6.5 Tastatura . . . 298
11.6.6 IEEE-488 standard . . . 300
11.6.7 Komodorova serijska veza (IEC) . . . 305
11.6.8 Priključak za proširenja . . . 308
11.6.9 Upravljački ulazi . . . 310
11.6.10 Korisnički priključak . . . 311
11.7 LOGIKA ZA UPRAVLJANJE MEMORIJOM . . . 312
11.8 LOGIKA ZA GENERISANJE TAKTOVA . . . 313
11.9 NAPAJANJE . . . 316

12 KONSTRUKCIJE . . . 318
12.1 CENTRONIKS INTERFEJS . . . 318
12.2 RS 232 INTERFEJS . . . 320
12.3 MODEM . . . 323
12.4 EPROM PROGRAMATOR . . . 324
12.5 ROM MODUL . . . 329

Dodatak . . . 331

Commodore za sva vremena

Obuhvata sve ono što je potrebno za razumevanje i korišćenje računara Commodore 64:

Osnovni pojmovi o računarima
Uvod u rad sa Commodore-om
BASIC
Simons BASIC
Mašinsko programiranje
Hardver
Konstrukcije
i još mnogo toga...

Commodore za sva vremena

Korisnicima Komodora izlažu se sve one informacije koje bi se inače morale tražiti u mnoštvu, za naše prilike, veoma skupih knjiga na stranim jezicima.

RAČUNARI

Kompletan rad o C–64. Na 330 strana je kazano mnogo toga što moramo znati o Komodoru.

MOJ MIKRO

Najkompletnija knjiga za C–64 koja se pojavila na svetskom tržištu.

SVET KOMPJUTERA



YU ISBN 86 80003 02 6